Высшее профессиональное образование

Л. Н. Ясницкий

ВВЕДЕНИЕ В ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

2-е издание

Учебное пособие



Информатика и вычислительная техника



УДК 519.767.4(075.8)
ББК 32.813я73
Я82

Рецензенты:
зав. кафедрой динамики и прочности машин Пермского государственного технического университета, д-р техн. наук, академик РАЕН, чл.-корр. АТН, проф. *Г.Л.Колмогоров*;
зав. кафедрой информатики Пермского государственного педагогического университета, д-р физ.-матем. наук, академик Международной академии информатизации, академик Академии информатизации образования РФ, проф. *Е.К.Хеннер*;
зав. кафедрой прикладной математики и информатики Пермского государственного университета, д-р физ.-матем. наук, академик Академии информатизации образования РФ, проф. *С.В.Русаков*

Ясницкий Л.Н.
Я82 Введение в искусственный интеллект : учеб. пособие для студ. высш. учеб. заведений / Л.Н.Ясницкий. — 2-е изд., испр. — М. : Издательский центр «Академия», 2008. — 176 с.
ISBN 978-5-7695-5390-5

Изложены два основных подхода, применяемые при создании систем искусственного интеллекта: технология экспертных систем и нейросетевые технологии. Освещены вопросы их практического использования при решении задач распознавания образов, прогнозирования, диагностики, оптимизации и т.д.

Рассмотрены проблемы применения интеллектуальных систем в экономике, бизнесе, финансах, машиностроении, политологии, медицине, криминалистике. Подробно описан новый раздел искусственного интеллекта, связанный с созданием интеллектуальных систем, имитирующих творческую деятельность математика-профессионала при аналитическом решении краевых задач математической физики.

Для студентов высших учебных заведений.

УДК 519.767.4(075.8)
ББК 32.813я73

ISBN 978-5-7695-5390-5

ПРЕДИСЛОВИЕ

Искусственный интеллект — это раздел информатики, посвященный моделированию интеллектуальной деятельности человека. Зародившийся более 700 лет назад в средневековой Испании искусственный интеллект оформился в самостоятельную научную область в середине XX в. Пройдя сложный, извилистый путь многократных метаний между чрезмерным оптимизмом и необоснованным скептицизмом, в наши дни искусственный интеллект получил блестящие практические приложения, открывающие перспективы, без которых немыслимо дальнейшее развитие цивилизации.

Методы искусственного интеллекта позволили создать эффективные компьютерные программы в самых разнообразных, ранее считавшихся недоступными для формализации и алгоритмизации, сферах человеческой деятельности, таких как медицина, биология, зоология, социология, культурология, политология, экономика, бизнес, криминалистика и т. п. Идеи обучения и самообучения компьютерных программ, накопления знаний, приемы обработки нечетких и неконкретных знаний позволили создать программы, творящие чудеса. Компьютеры успешно борются за звание чемпиона мира по шахматам, моделируют творческую деятельность человека, создавая музыкальные и поэтические произведения, распознают образы и сцены, распознают, понимают и обрабатывают речь, тексты на естественном человеческом языке. Нейрокомпьютеры, созданные по образу и подобию человеческого мозга, успешно справляются с управлением сложными техническими объектами, диагностикой заболеваний человека, неисправностей сложных технических устройств; предсказывают погоду и курсы валют, результаты голосований; выявляют хакеров и потенциальных банкротов; помогают абитуриентам правильно выбрать специальность и т. д.

Мы уже привыкли к тому, что компьютеры «умнеют» буквально на глазах, а компьютерные программы становятся все более и более интеллектуальными. Само по себе понятие интеллекта постоянно претерпевает изменения по мере развития науки и человека. Давно уже не считаются интеллектуальными задачи, состоящие в выполнении арифметических операций сложения, умножения, деления. Не считается интеллектуальной задача интегри-

рования дифференциального уравнения, если для нее известен строго детерминированный алгоритм. В настоящее время принято считать интеллектуальными задачи, которые на современном этапе не поддаются алгоритмизации в традиционном смысле этого слова. Это задачи, для решения которых требуются манипуляции с нечеткими, неконкретными, ненадежными, расплывчатыми и даже нетрадиционными знаниями.

Каким же образом удается решать такие задачи?

Автор попытался ответить на этот вопрос, собрав и проанализировав методы искусственного интеллекта, применяющиеся в различных разделах и направлениях. Основной упор он сделал на то, что принципиально отличает интеллектуальные системы. Это, в первую очередь, возможность их обучения, накопления знаний во время работы компьютерных программ, способность самообучения, самоорганизации, самосовершенствования.

В книге нет готовых алгоритмов и программ. Однако читатель, владеющий современными инструментальными средствами информатики, может реализовать излагаемые идеи и методы в своей практической деятельности. Кроме того, знание теоретических основ искусственного интеллекта полезно при освоении современных интеллектуальных пакетов прикладных программ, число которых растет ускоряющимися темпами.

Книга содержит изложение двух основных подходов, применяемых при создании систем искусственного интеллекта, — технологии экспертных систем и нейросетевых технологий. В некотором смысле эти подходы являются альтернативными. Первый из них предполагает создание базы знаний о предметной области и механизма, обрабатывающего эти знания с целью получения полезного логического вывода. Согласно второму подходу знания хранятся и обрабатываются в неявной форме подобно тому, как это происходит в человеческом мозге. Часто эти подходы конкурируют между собой, поэтому при проектировании систем искусственного интеллекта важно сделать правильный выбор.

В книге большое внимание уделяется вопросам практического применения методов искусственного интеллекта, в частности, при решении задач распознавания образов, прогнозирования, диагностики, оптимизации, при моделировании творческой деятельности человека, создании игровых компьютерных программ. Приведены принципы построения, общее описание и опыт применения систем искусственного интеллекта, используемых в промышленности, бизнесе, экономике, медицине, криминалистике, политологии, педагогике.

Последняя глава книги, названная «Интеллектуальное математическое моделирование», по своей сути открывает новое направление искусственного интеллекта. Здесь автор изложил свой опыт по созданию интеллектуальной системы, имитирующей твор-

ческую деятельность математика-профессионала, его интуицию и опыт, необходимые при аналитическом решении краевых задач математической физики.

Таким образом, предлагаемое издание охватывает весьма широкий круг вопросов и несмотря на учебную направленность содержит элементы научной монографии. Поэтому автор рекомендует книгу как студентам, так и зрелым программистам, аспирантам и ученым, посвятившим себя этой увлекательной научной области.

Автор использовал рукописные материалы из библиотеки Юрия Владимировича Девингталя, подаренной Пермскому государственному университету Валентиной Васильевной Девингталь. Кроме того, в книге приводится информация, любезно предоставленная автору специалистами в области искусственного интеллекта: Р. П. Абусевым, И. А. Грибановым, В. А. Игошиным, В. А. Краснобаевым, М. А. Марценюком, С. И. Чуприной. Результаты решения краевых задач гл. 7 получены совместно с С. Л. Гладким и Ф. Г. Салахутдиновым. Экспертная система интеллектуального математического моделирования разработана совместно с С. Л. Гладким, а ее первоначальное тестирование выполнено А. В. Семеновой, Ф. Г. Салахутдиновым, А. В. Тарантиной, О. А. Кулинской. Принципы построения нейросетевой системы диагностики авиационных двигателей разработаны совместно с начальником отдела диагностики АО «Авиадвигатель» В. Ф. Халиуллиным, а принципиальные основы создания системы кардиодиагностики разрабатывались с врачом-консультантом Пермского кардиологического центра Ю. К. Филоненко. Принципы построения нейросетевого детектора лжи обсуждались с полковником МВД РФ А. М. Петровым.

Всем своим коллегам, предоставившим информацию, принявшим участие в разработке систем искусственного интеллекта и в подготовке рукописи книги, автор выражает искреннюю благодарность.

ГЛАВА 1
ПРОШЛОЕ, НАСТОЯЩЕЕ И БУДУЩЕЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

1.1. ИСТОРИЧЕСКИЙ ОЧЕРК

Человек — это самый сложный из доступных для нашего восприятия объект, а способность мышления — его главное свойство — атрибут. Искусственный интеллект — наука, поставившая своей целью изучение и моделирование атрибута человека. Какова природа мышления? Какие процессы происходят в нашем организме, когда мы думаем, чувствуем, видим, понимаем? Возможно ли в принципе понять, как работает наш мозг, и заставить мыслить неживую природу? На протяжении тысячелетий человек задавался этими вопросами, но до сих пор мы не можем на них ответить с полной определенностью.

История попыток создания искусственного подобия человеческого разума насчитывает более 700 лет. Первую зафиксированную в истории попытку создания машины, моделирующей человеческий разум, связывают с именем испанского рыцаря, поэта, философа, богослова, алхимика, изобретателя Раймунда Луллия.

Представляет огромный интерес сама личность этого человека. Любимец короля, дуэлянт и повеса, как о нем пишут историки, рыцарь Раймунд Луллий вдруг отказывается от светских развлечений и идет в монастырь, чтобы стать мудрецом. Его решение вызвано весьма благородной целью — постичь науки и с их помощью излечить от болезни свою даму сердца. К сожалению, истории не известно, удалось ли Луллию достичь своей цели. Известно только, что в возрасте 80 лет он был насмерть забит камнями. Это случилось при неудачной попытке чтения лекций по искусственному интеллекту.

Луллий родился в 1235 и умер в 1315 г. В его времена ученые были заняты поиском неких универсальных понятий и истин, которые, будучи связанными между собой, давали бы общую картину мироздания, а значит, и ответы на все интересующие человечество вопросы. Это был век философов-мудрецов, астрологов и алхимиков, занятых поисками философского камня.

Развивая традиции ученых своего времени, Луллий сконструировал машину, состоявшую из системы кругов, имевших возможность вращаться. Каждый круг был поделен на секторы, окрашенные в разные цвета и помеченные латинскими буквами. Круги

соединялись друг с другом, и, приводя их во вращение, можно было получить различные сочетания символов и цветов — так называемую формулу истины.

Машины Луллия могли работать в различных предметных областях и давать ответы на всевозможные вопросы, составлять гороскопы, ставить диагнозы болезней, делать прогнозы на урожай. В наиболее позднем варианте машина Луллия состояла из 14 кругов, размеченных буквами и раскрашенных в различные цвета, которые символизировали различные понятия, элементы, стихии, субъекты и объекты знания. Круги приводились в движение системой рычагов. Поворачиваясь, они могли образовывать около 18 квадриллионов ($18 \cdot 10^{15}$) разнообразных сочетаний буквенных и цветовых «истин». Запросы в машину вводились с помощью поворота внутреннего круга, на котором было начертано девять вариантов вопросов: Что? Почему? Из чего? Сколько? Каким образом? Где? Когда? Какое? Которое из двух?

Выражаясь современным языком, машина Луллия, по существу, представляла собой механическую экспертную систему, наделенную базой знаний, устройствами ввода и вывода, естественным языком общения. Свести к логическим операциям если не все знания о мире, то хотя бы часть из них, а затем поручить не человеческому мозгу, а механическому устройству процедуру вывода «формул знания», следующих из накопленной базы знаний, — эта идея искусственного интеллекта, впервые высказанная и реализованная средневековым рыцарем Раймундом Луллием, прожила семь веков и достигла в наши дни своего расцвета и триумфа.

В 40-х годах XX в. с появлением электронно-вычислительных машин искусственный интеллект обрел второе рождение. Произошло выделение искусственного интеллекта в самостоятельное научное направление. Сам термин «искусственный интеллект» (artifical intelligence) был предложен в 1956 г. на семинаре с аналогичным названием в Станфордском университете (США).

С тех пор история искусственного интеллекта представляла собой постоянные споры и метания между двумя крайностями — оптимизмом и пессимизмом. Интересны знаменитые предсказания американского экономиста и социолога, исследователя в области теории управления, моделирования социальных процессов Г. Саймона, сделанные в 1957 г. Приведем некоторые из них:

в ближайшее десятилетие ЭВМ завоюет титул чемпиона мира по шахматам;

в пределах десяти лет ЭВМ откроет и сумеет доказать важную новую математическую теорему;

в десятилетний срок большинство теорий в области психологии примет вид программ для вычислительной машины.

Сейчас, спустя почти полвека, мы видим, что предсказания Саймона постепенно сбываются, что он ошибался только в сро-

ках. Мы также можем отметить, что эйфория вокруг молодой кибернетики имела как положительные, так и отрицательные последствия. С одной стороны, она стимулировала интерес общественности к новому научному направлению, выразившийся в выделении крупных грантов правительством США. С другой стороны, кибернетика стала объектом весьма резкой критики более «трезво мыслящих» ученых. Мы также знаем, к каким тяжелым последствиям привела эта критика в СССР, когда за решение проблем научных дискуссий взялся государственный репрессивный аппарат.

Вскоре после признания искусственного интеллекта самостоятельной отраслью науки произошло его разделение на два основных направления: *нейрокибернетику* и *кибернетику «черного ящика»*. Первое из этих направлений иногда называют *низкоуровневым*, или *восходящим*, а второе — *высокоуровневым*, или *нисходящим*.

Основную идею нейрокибернетики можно сформулировать следующим образом. Единственный объект, способный мыслить, — это человеческий мозг. Поэтому любое мыслящее устройство должно быть обязательно выполнено по образу и подобию человеческого мозга, воспроизводить его структуру, его принцип действия. Таким образом, нейрокибернетика занимается аппаратным моделированием структуры мозга и его деятельности.

Как известно, мозг человека состоит из большого количества взаимосвязанных нервных клеток — нейронов. Поэтому усилия нейрокибернетиков сосредоточены на разработке элементов, подобных нейронам, и объединении этих элементов в системы — нейросети и нейрокомпьютеры. Первые нейросети и нейрокомпьютеры были предложены и созданы американскими учеными В. Мак-Каллоком, В. Питтсом и Ф. Розенблаттом в конце 1950-х годов. Это были устройства, моделирующие человеческий глаз и его взаимодействие с мозгом. Устройства умели распознавать буквы алфавита, однако были чувствительны к их написанию.

Сегодня нейрокомпьютерные и нейросетевые технологии являются одним из наиболее перспективных и быстро развивающихся разделов искусственного интеллекта. Крупных успехов в этой области добились японские исследователи. Ими создан компьютер VI поколения — нейрокомпьютер, моделирующий структуру мозга и имеющий обширную базу знаний. Значительных успехов в этой области добились российские ученые. Отечественные нейрокомпьютеры уже давно применяются для управления сложными техническими объектами военного назначения.

В отличие от нейрокибернетики кибернетика «черного ящика» не придает значения принципу действия мыслящего устройства. Главное, чтобы оно адекватно моделировало его функциональную деятельность. Это направление искусственного интеллекта

ориентировано на поиски алгоритмов решения интеллектуальных задач с использованием существующих компьютеров независимо от их аппаратной базы.

Поставив перед собой задачу моделирования функций мозга, ученые столкнулись с серьезной проблемой. Оказалось, что несмотря на многовековую историю исследований ни одна из существующих наук (философия, психология, лингвистика и др.) не смогла предложить сколько-нибудь конкретный алгоритм человеческого мышления. Поэтому кибернетикам пришлось создавать собственные модели мышления.

В конце 50-х гг. XX в. появилась модель лабиринтного поиска. Согласно этому подходу решение интеллектуальной задачи выполнялось путем перебора огромного количества вариантов, который представлялся в виде движения по лабиринту. Создание таких алгоритмов, по словам их критиков, было не более разумно, чем попытки заново написать все книги, хранящиеся в Британском музее, посадив за пишущие машинки обезьян и надеясь, что обезьяны рано или поздно чисто случайно сумеют напечатать осмысленное слово, фразу или страницу. В настоящее время модель лабиринтного поиска признается тупиковой и имеет ограниченное использование в игровых компьютерных программах.

В начале 1960-х гг. началась эпоха эвристического программирования. Как писал автор этого термина американский математик Пойа, цель эвристики — исследовать методы и правила, как делать открытия и изобретения. Это очень сложная проблема. Дело в том, что Архимед, выпрыгнувший из ванны с криком «Эврика!», не объяснил, каким образом он догадался, что тело, погруженное в жидкость, теряет в своем весе ровно столько, сколько весит вытесненный им объем воды. Ньютон открыл закон всемирного тяготения, наблюдая за падением яблока. Менделеев пришел к принципу построения периодической таблицы во сне. Поэтов и музыкантов вдохновляют к творческим поискам возвышенные чувства, разобраться в которых в принципе не возможно.

Чтобы понять механизмы творческого мышления, авторы эвристического подхода провели эксперимент. Была отобрана группа студентов, не знакомых с математической логикой. Каждый студент должен был доказать самостоятельно одну или несколько теорем из учебника, не заглядывая в него. При этом ему вменялось в обязанность рассуждать вслух, делать любые записи, прекращать работу, если становилось ясно, что выбран неверный путь, и начинать все сначала.

Обработав магнитофонные записи, выкладки, черновики студентов, программисты нашли эвристики — способы, которыми пользовались студенты, доказывая теоремы. А затем с помощью этих эвристик была составлена программа, известная под назва-

нием «Логик-теоретик», которую принято считать родоначальницей эвристического программирования. И эта программа доказала все теоремы, какие были в учебнике, и сформулировала дополнительно те, которых не хватало до полной логической завершенности курса.

Наряду с указанными выше двумя подходами к проблеме моделирования мышления и создания искусственного интеллекта существует третий, названный эволюционным программированием (моделированием). Смысл этого подхода состоит в том, что процесс моделирования человека заменяется моделированием процесса его эволюции.

Серьезный прорыв в практических приложениях искусственного интеллекта произошел в середине 1970-х гг., когда, отказавшись от поисков универсального алгоритма мышления, программисты начали моделировать конкретные знания специалистов-экспертов. Открылось новое направление искусственного интеллекта — экспертные системы. С появлением экспертных систем бизнес в сфере интеллектуальных информационных технологий впервые становится рентабельным.

С середины 1980-х гг. искусственный интеллект — это одно из наиболее привлекательных в коммерческом отношении направлений компьютерной индустрии. Растут ежегодные капиталовложения, создаются промышленные и военные экспертные системы. В качестве альтернативы экспертным системам появляются и успешно завоевывают рынок нейросетевые и нейрокомпьютерные технологии, в которых, подобно процессам, происходящим в мозгу, знания растворяются в межнейронных связях, а процесс программирования системы заменяется ее обучением.

1.2. НАПРАВЛЕНИЯ РАЗВИТИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Сегодня искусственный интеллект — это обширная область исследований и разработок интеллектуальных систем, предназначенных для работы в трудно формализуемых областях деятельности человека. Для задач, решаемых методами искусственного интеллекта, характерно наличие большого числа степеней свободы с числом вариантов поиска решений, приближающимся к бесконечности. В отличие от жестко детерминированных компьютерных программ системы искусственного интеллекта сами ищут пути решения поставленной задачи. При этом они могут менять свои параметры и структуру, совершенствоваться и развиваться, жить самостоятельной, не зависящей от воли разработчика жизнью.

Разработка интеллектуальных систем, основанных на знаниях. До недавнего времени это направление считалось основным и

наиболее плодотворным в развитии искусственного интеллекта. Оно связано с разработкой моделей представления знаний, созданием баз знаний, образующих ядро экспертных систем.

Нейросетевые и нейрокомпьютерные технологии. Это направление является альтернативным предыдущему как в идеологическом, так и в практическом плане. Искусственные нейронные сети и нейрокомпьютеры в значительной мере заимствуют принципы работы головного мозга. Знания в них не отделены от процессора, а равномерно распределены и существуют неявно в виде сил синаптических связей. Такие знания не закладываются изначально, а приобретаются в процессе обучения.

Распознавание образов. К распознаванию образов в искусственном интеллекте относят широкий круг проблем: распознавание изображений, символов, текстов, запахов, звуков, шумов. На рынке программных средств имеются системы, основанные на распознавании по признакам, оснащенные базами данных и знаний, имеющих возможность адаптации и обучения. Однако в последнее время становятся популярными гибридные системы, в которых наряду с технологиями экспертных систем используются и нейросетевые технологии.

Игры и творчество. Традиционно искусственный интеллект включает в себя интеллектуальные задачи, решаемые при игре в шахматы, шашки, го, каллах. В основе этого направления лежит один из ранних подходов — лабиринтная модель плюс эвристики. Кроме того, в современных программах-игроках наиболее полно удалось реализовать центральную идею искусственного интеллекта — обучение, самообучение и самоорганизацию.

В широком смысле слова под игрой понимается некая конфликтная ситуация, участники которой своими действиями не только достигают своих личных целей, но и влияют на достижимость целей другими участниками игры. Ясно, что под такое толкование игры подпадают многие экономические, политические и военные конфликты.

Компьютерное творчество представляет пока чисто теоретический интерес. Наибольший прогресс достигнут в сочинении компьютерной музыки. Разработаны различные модели художественного и поэтического творчества, имеющие больше познавательный, чем практический интерес.

Компьютерная лингвистика. Начиная с 50-х гг. XX в. и по настоящее время одной из популярных тем исследований искусственного интеллекта является область машинного перевода. Первая программа в этой области — переводчик с английского языка на русский. Первая идея — пословный перевод. В настоящее время используются более сложные структуры естественно-языковых интерфейсов, которые включают в себя:

морфологический анализ — анализ слов в тексте;

синтаксический анализ — анализ предложений, грамматики и связей между словами;

семантический анализ — анализ смысла каждого предложения на основе базы знаний, на которую ориентирована конкретная программа-переводчик;

прагматический анализ — анализ смысла предложений в окружающем контексте с помощью базы знаний.

Другой проблемой компьютерной лингвистики является разработка естественно-языкового интерфейса между человеком и машиной. Здесь немаловажную роль могут сыграть нейросетевые технологии, с помощью которых удается научить компьютер правильному произношению слов. В проектах создания компьютеров V и VI поколений решению этой проблемы уделено первостепенное внимание.

Интеллектуальные роботы. Роботы — это технические устройства, предназначенные для автоматизации человеческого труда. Само слово «робот» появилось в 20-х гг. XX в. Его автор — чешский писатель Карел Чапек.

В настоящее время в промышленности применяется огромное количество роботов-манипуляторов, работающих по жесткой схеме управления. В отличие от них интеллектуальные роботы обладают способностью самообучаться и самоорганизовываться, адаптироваться к изменяющейся окружающей обстановке.

Компьютерные вирусы. Сегодня трудно назвать компьютерного пользователя, избежавшего знакомства с этим видом программной продукции.

Последние поколения вирусов обладают всеми атрибутами систем искусственного интеллекта. Они свободно перемещаются по компьютерам, мутируют и размножаются, обучаются, меняют свои параметры и структуру.

Воздействие компьютерных вирусов значительно возросло с появлением сети Internet. По прогнозам специалистов, неприятности, которые мы испытываем сегодня, представляются ничтожными по сравнению с теми перспективами, которые ожидают нас с проникновением компьютерных вирусов в сферу интеллектуальных роботов.

Интеллектуальное математическое моделирование. Это компьютерное математическое моделирование с использованием методов искусственного интеллекта.

Интеллектуальные системы подобного рода имитируют творческую деятельность математика-профессионала, занимающегося решением краевых задач математической физики. Они обладают базами знаний, содержащими нужные теоремы, математические зависимости и эвристические правила, обобщающие опыт и интуицию математика-профессионала, способны к обучению с помощью учителя и к самообучению.

Контрольные вопросы

1. Опишите назначение и принцип действия машины Р. Луллия.
2. В чем суть модели лабиринтного поиска и эвристического метода?
3. Чем отличаются нейрокибернетические методы от методов кибернетики «черного ящика»?
4. В чем смысл терминов «восходящее» и «нисходящее» направления искусственного интеллекта?
5. Что такое эволюционное программирование?
6. Перечислите основные направления искусственного интеллекта.
7. Что такое интеллектуальное математическое моделирование?

ГЛАВА 2

СИСТЕМЫ, ОСНОВАННЫЕ НА ЗНАНИЯХ

2.1. ДАННЫЕ И ЗНАНИЯ

При изучении искусственного интеллекта естественно возникает вопрос: «Что такое знания и чем они отличаются от данных?». Приведем определения, заимствованные из учебника информатики [13].

Данные — это отдельные факты, характеризующие объекты, процессы и явления предметной области, а также их свойства.

При обработке на ЭВМ данные трансформируются, последовательно проходя следующие этапы:

данные, существующие как результат измерений и наблюдений;

данные на материальных носителях информации — в таблицах, протоколах, справочниках;

структуры данных в виде диаграмм, графиков, функций;

данные в компьютере на языке описания данных;

базы данных.

Знания связаны с данными, основываются на них, но представляют собой результат мыслительной деятельности человека, обобщают его опыт, полученный в ходе практической деятельности. Знания — это выявленные закономерности предметной области [13].

При обработке на ЭВМ знания трансформируются аналогично данным:

знания, существующие в памяти человека как результат обучения, воспитания, мышления;

знания, помещенные на материальных носителях — учебниках, инструкциях, методических пособиях, книгах;

знания, описанные на языках представления знаний и помещенные в компьютер;

базы знаний.

Для хранения данных используются базы данных. Для них характерны большой объем и относительно небольшая стоимость информации. Для хранения знаний используются базы знаний. Они, наоборот, отличаются сравнительно небольшими объемами, но исключительно дорогими информационными массивами.

Знания могут быть классифицированы на *поверхностные* — знания о видимых взаимосвязях между отдельными событиями и фак-

тами в предметной области, и *глубинные* — абстракции, аналогии, схемы, отображающие структуру и процессы в предметной области.

Кроме того, знания можно разделить на *процедурные* и *декларативные*.

Исторически первичными были процедурные знания, т.е. знания, растворенные в алгоритмах. Они управляли данными. Для их изменения требовалось изменять программы.

Рассмотрим, например, фрагмент программы на Паскале.

```
Pi:= 3.14;
R:= 20;
S:= Pi * R * R;
WRITELN ('Площадь круга S =', S).
```

Первые два оператора представляют собой данные, третий оператор — знание. Оно является результатом интеллектуальной деятельности древних геометров и представляет собой закон, выражающий площадь круга через его радиус.

Однако с развитием искусственного интеллекта приоритет данных постепенно изменялся и все бóльшая часть знаний сосредоточивалась в структурах данных, т.е. увеличивалась роль декларативных знаний.

Существуют десятки способов представления декларативных знаний для различных предметных областей. Большинство из них может быть сведено к следующим классам: продукционные; фреймы; семантические сети.

2.2. МЕТОДЫ ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

2.2.1. Продукционные правила

Продукционная система состоит из трех основных компонентов, схематично изображенных на рис. 2.1. Первый из них — это база правил типа ЕСЛИ (условие), ТО (действие): ЕСЛИ холодно, ТО надеть шубу; ЕСЛИ идет дождь, ТО взять зонтик, и т.п.

Вторым компонентом является рабочая память, в которой хранятся исходные данные к задаче и выводы, полученные в ходе работы системы.

Третий компонент — механизм логического вывода, использующий правила в соответствии с содержимым рабочей памяти.

Рассмотрим конкретный пример. В базе правил экспертной системы имеются два правила.

Правило 1: ЕСЛИ «намерение — отдых» и «дорога ухабистая», ТО «использовать джип».

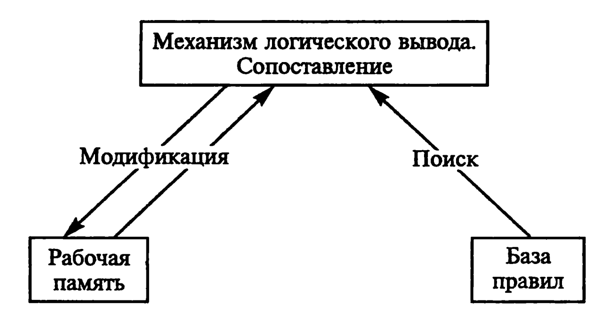


Рис. 2.1. Блок-схема продукционной системы

Правило 2: ЕСЛИ «место отдыха — горы», ТО «дорога ухабистая».

Допустим, что в рабочую память поступили исходные данные: «намерение — отдых»; «место отдыха — горы».

Механизм вывода начинает сопоставлять образцы из условных частей правил с образцами, хранимыми в рабочей памяти. Если образцы из условной части имеются в рабочей памяти, то условная часть считается истинной, в противном случае — ложной.

В данном примере при рассмотрении правила 1 оказывается, что образец «намерение — отдых» имеется в рабочей памяти, а образец «дорога ухабистая» отсутствует, поэтому условная часть правила 1 считается ложной. При рассмотрении правила 2 выясняется, что его условная часть истинна. Механизм вывода выполняет заключительную часть этого правила, и образец «дорога ухабистая» заносится в рабочую память. Правило 2 при этом выбывает из числа кандидатов на рассмотрение.

Снова рассматривается правило 1, условная часть которого теперь становится истинной, и содержимое рабочей памяти пополняется образцом «использовать джип». В итоге правил, которые можно было бы применять, не остается и система останавливается.

В рассмотренном примере приведен прямой вывод — от данных к поиску цели. Однако применяют и обратный вывод — от цели для ее подтверждения к данным. Продемонстрируем этот способ на нашем примере. Допустим, что наряду с исходными данными «намерения — отдых»; «место отдыха — горы» имеется цель «использовать джип».

Согласно правилу 1 для достижения этой цели требуется выполнение условия «дорога ухабистая», поэтому условие становится новой целью. При рассмотрении правила 2 оказывается, что условная часть этого правила в данный момент истинна, поэтому рабочая память пополняется образцом «дорога ухабистая». При

повторном рассмотрении правила 1 подтверждается цель «использовать джип».

При обратном выводе система останавливается в двух случаях: либо достигается первоначальная цель, либо кончаются правила. При прямом выводе система останавливается только тогда, когда кончаются правила, либо при появлении в рабочей памяти специально предусмотренного образца, например, «использовать джип».

В приведенном примере на каждом этапе прямого вывода можно было использовать только одно правило. В общем же случае на каждом этапе вывода таких правил несколько, и тут возникает проблема выбора. Например, введем в рассмотрение еще одно правило.

Правило 3: ЕСЛИ «намерение — отдых», ТО «нужна скорость».

Кроме того, введем условие останова системы — появление в рабочей памяти образца «использовать джип».

Теперь на первом этапе прямого вывода появляется возможность применять либо правило 2, либо правило 3. Если сначала применить правило 2, то на следующем этапе можно будет применять правило 1 и правило 3. Если на этом этапе применить правило 1, то выполнится условие останова системы, но если прежде применить правило 3, то потребуется еще один этап вывода.

Этот пример показывает, что выбор применяемого правила оказывает прямое влияние на эффективность вывода. В реальной системе, где имеется множество правил, появляется проблема их оптимального выбора.

Если на каждом этапе логического вывода существует множество применимых правил, то это множество носит название *конфликтного набора*, а выбор одного из них называется *разрешением конфликта.*

Аналогичная ситуация возникает и при обратном выводе. Например, дополним предыдущий пример еще одним правилом.

Правило 4: ЕСЛИ «место отдыха — пляж», ТО «дорога ухабистая».

Если на основании этого условия подтверждается цель «использовать джип», то для достижения первоначальной цели достаточно применить только одно правило 1, однако, чтобы подтвердить новую цель «дорога ухабистая», открывается возможность применения правила 1, нужно использовать либо правило 2, либо правило 4. Если сначала применить правило 2, то это будет самый удачный выбор, поскольку сразу же можно применить и правило 1. С другой стороны, если попытаться применить правило 2, то, поскольку образца «место отдыха — пляж», который является условием правила 4, в рабочей памяти не существует и, кроме того, не существует правила, подтверждающего его, данный выбор является неудачным. И лишь со второго захода, применяя правило 2, можно подтвердить цель «дорога ухабистая».

Следует обратить внимание на то, что при обратном выводе правило 3, которое не оказывает прямого влияния на достижение цели, не принималось в расчет с самого начала. Таким образом, для обратных выводов характерна тенденция исключения из рассмотрения правил, не имеющих прямого отношения к заданной цели, что позволяет повысить эффективность вывода.

Продукционная модель — это наиболее часто используемый способ представления знаний в современных экспертных системах. Основными преимуществами продукционной модели являются наглядность, высокая модульность, легкость внесения изменений и дополнений, простота механизма логического вывода.

2.2.2. Фреймы

В психологии и философии используется понятие абстрактного образа. Например, слово «автомобиль» вызывает у слушающих образ устройства, способного перемещаться, имеющего четыре колеса, салон для шофера и пассажиров, двигатель, руль. Приведенное описание абстрактного образа «автомобиль» является минимальным и из него ничего нельзя убрать без потери его сущности.

Фрейм — это модель абстрактного образа, минимально возможное описание сущности какого-либо объекта, явления, события, ситуации, процесса. Фрейм состоит из имени и отдельных единиц, называемых слотами. Он имеет однородную структуру:

ИМЯ ФРЕЙМА
Имя 1-го слота: значение 1-го слота
Имя 2-го слота: значение 2-го слота
....
Имя N-го слота: значение N-го слота.

В качестве значения слота может выступать имя другого фрейма. Таким образом фреймы объединяются в сеть. Свойства фреймов наследуются сверху вниз, т.е. от вышестоящих к нижестоящим через АКО-связи (начальные буквы английских слов «A Kind Of», что можно перевести как «это»). Слот с именем АКО указывает на имя фрейма более высокого уровня иерархии.

Например, на рис. 2.2 фрейм «Студент» имеет ссылки на вышестоящие фреймы: «Человек» и «Млекопитающее». Поэтому на вопрос: «Может ли студент мыслить?» — ответ будет положительным, так как этим свойством обладает вышестоящий фрейм «Человек».

Если одно и тоже свойство указывается в нескольких связанных между собой фреймах, то приоритет отдается нижестоящему фрейму. Так, возраст фрейма «Студент» не наследуется из вышестоящих фреймов.

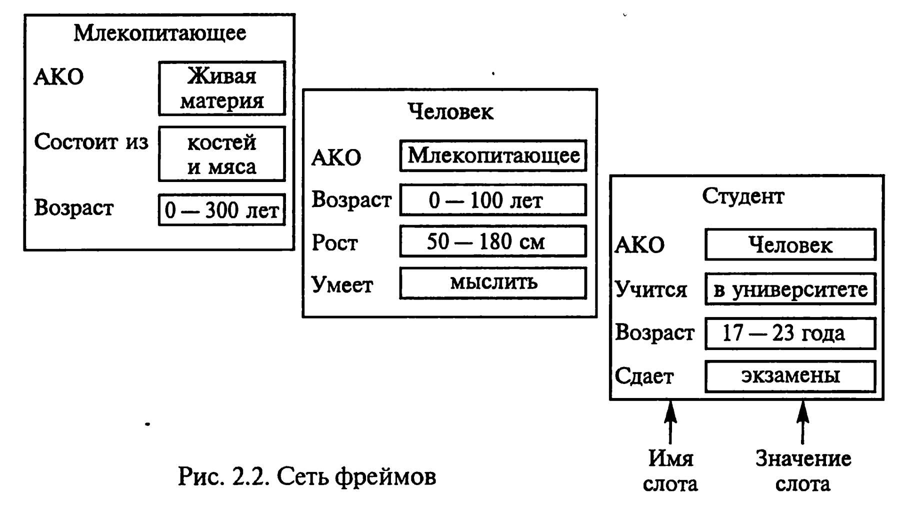


Рис. 2.2. Сеть фреймов

Основным преимуществом фреймов как способа представления знаний является наглядность и гибкость в употреблении. Кроме того, фреймовая структура согласуется с современными представлениями о хранении информации в памяти человека.

2.2.3. Семантические сети

В основе этого способа представления знаний лежит идея о том, что любые знания можно представить в виде совокупности *понятий* (объектов) и *отношений* (связей). Семантическая сеть представляет собой ориентированный граф, вершинами которого являются понятия, а дугами — отношения между ними. Сам термин «семантическая» означает смысловая.

Пример семантической сети приведен на рис. 2.3.

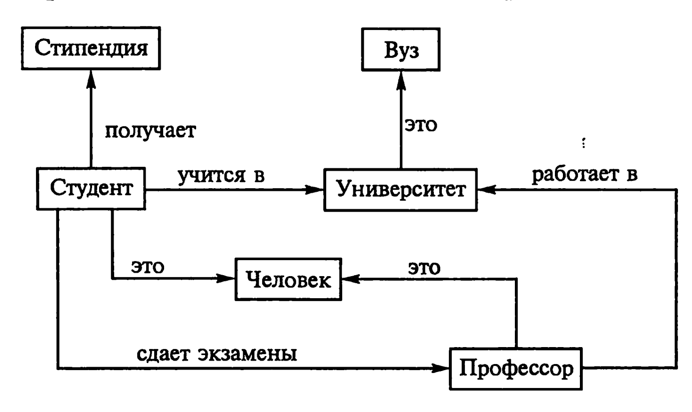


Рис. 2.3. Семантическая сеть

Основным преимуществом этой модели является наглядность представления знаний, а также соответствие современным представлениям об организации долговременной памяти человека. Недостаток — сложность поиска вывода, а также сложность корректировки, т.е. удаления и дополнения сети новыми знаниями.

2.3. ЭКСПЕРТНЫЕ СИСТЕМЫ

2.3.1. Предметные области

Знания, которыми обладает специалист в какой-либо области, можно разделить на формализуемые и плохо формализуемые. Формализуемые знания излагаются в книгах и руководствах в виде законов, формул, моделей, алгоритмов. Формализуемые знания характерны для точных наук, таких как математика, физика, химия, астрономия. Науки, которые принято называть описательными, обычно оперируют с плохо формализуемыми знаниями. К таким наукам можно отнести, например, зоологию, ботанику, экологию, социологию, педагогику, медицину и др.

Существуют неформализуемые знания, которые вообще не попадают в книги и руководства в связи с их неконкретностью, субъективностью, приблизительностью. Знания этого рода являются результатом многолетних наблюдений, опыта работы, интуиции. Они обычно представляют собой множество эмпирических и эвристических приемов и правил. Такие знания передаются из поколения в поколение в виде определенных навыков, ноу-хау, секретов ремесла. Есть также знания, которые не могут быть выражены ни в математическом виде, ни в терминах обычного человеческого языка. Такими знаниями обладают религиозные деятели, экстрасенсы, контактеры, шаманы.

Класс задач, относящихся к неформализуемым и плохо формализуемым знаниям, значительно больше класса задач, для которых знания формализуемы. Этим объясняется особая популярность и широкое практическое применение экспертных систем, которые открыли возможность применения компьютерных технологий в предметных областях, в которых знания плохо формализуемы.

2.3.2. Обобщенная структура

Экспертные системы — это сложные программные комплексы, аккумулирующие знания специалистов в конкретных предметных областях и тиражирующие эти знания для консультаций менее квалифицированных пользователей.

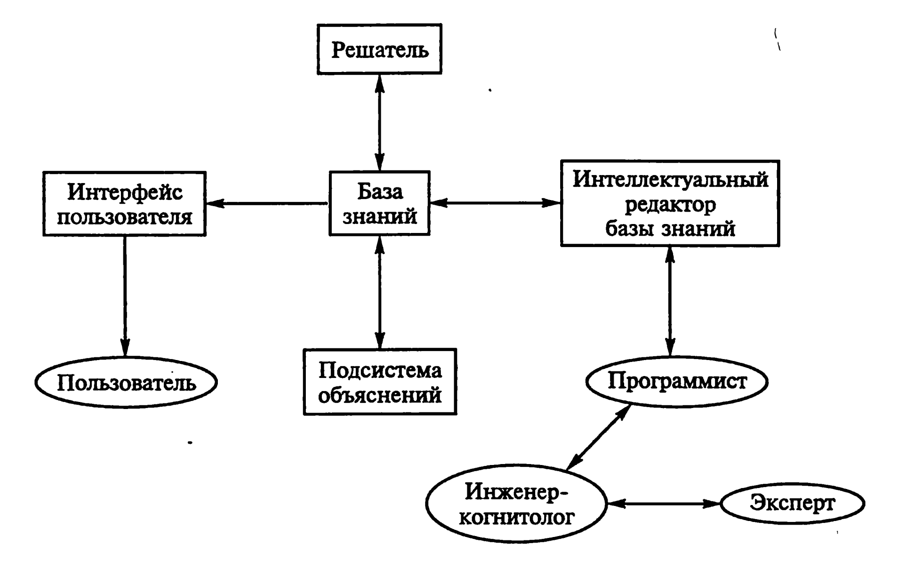


Рис. 2.4. Типичная блок-схема экспертной системы

Обобщенная блок-схема экспертной системы представлена на рис. 2.4. Обычно в ее состав входят следующие взаимосвязанные между собой модули:

база знаний — ядро экспертной системы, совокупность знаний предметной области, записанная на машинном носителе в форме, понятной эксперту и пользователю;

интеллектуальный редактор базы знаний — программа, представляющая инженеру-когнитологу и программисту возможность создавать базу знаний в диалоговом режиме. Она включает в себя системы вложенных меню, шаблонов языка представления знаний, подсказок (help-режим) и других сервисных средств, облегчающих работу с базой знаний;

интерфейс пользователя — комплекс программ, реализующих диалог пользователя с экспертной системой на стадии как ввода информации, так и получения результатов;

решатель (синонимы: *дедуктивная машина*, *блок логического вывода*) — программа, моделирующая ход рассуждений эксперта на основании знаний, имеющихся в базе знаний;

подсистема объяснений — программа, позволяющая пользователю получать ответы на вопросы: «Как была получена та или иная рекомендация?» и «Почему система приняла такое решение?». Ответ на вопрос «Как?» — это трассировка всего процесса получения решения с указанием исполняющих фрагментов базы знаний, т.е. всех шагов цепи умозаключений. Ответ на вопрос «Почему?» — ссылка на умозаключение, непосредственно предшествовавшее полученному решению, т.е. отход на один шаг назад.

В коллектив разработчиков экспертной системы входят как минимум четыре специалиста (или четыре группы специалистов): эксперт, инженер-когнитолог, программист, пользователь. Возглавляет коллектив инженер-когнитолог — ключевая фигура при разработке систем, основанных на знаниях. Обычно это руководитель проекта, в задачу которого входит организация всего процесса создания экспертной системы. С одной стороны, он должен быть специалистом в области искусственного интеллекта, а с другой — разбираться в предметной области, общаться с экспертом, извлекая и формализуя его знания, передавать их программисту, кодирующему и помещающему их в базу знаний экспертной системы.

Экспертная система работает в двух режимах — приобретения знаний и решения задач или консультаций.

В режиме приобретения знаний происходит формирование базы знаний. В режиме решения задач общение с экспертной системой осуществляет конечный пользователь.

Обычно знания, которыми располагает эксперт, различаются степенью надежности, важности, четкости. В этом случае они снабжаются некоторыми весовыми коэффициентами, которые называют *коэффициентами доверия*. Такие знания обрабатываются с помощью алгоритмов нечеткой математики.

В процессе опытной эксплуатации коэффициенты доверия могут подвергаться корректировке. В этом случае говорят, что происходит обучение экспертной системы. Процесс обучения экспертной системы может производиться автоматически с помощью обучающего алгоритма либо путем вмешательства инженера-когнитолога, выполняющего роль учителя.

2.3.3. Этапы и технология разработки

В процессе разработки экспертные системы проходят определенные стадии, в результате которых создаются различные версии, называемые прототипами:

демонстрационный прототип — экспертная система, которая решает часть требуемых задач, демонстрируя жизнеспособность метода инженерии знаний. Работает, имея в базе знаний всего 50...100 правил. Время разработки такой экспертной системы — 6...12 мес.;

исследовательский прототип — экспертная система, которая решает все требуемые задачи, но неустойчива в работе и неполностью проверена. База знаний содержит 200...500 правил. Разработка занимает 3...6 мес.;

действующий прототип — надежно решает все задачи, но для решения сложных задач может потребоваться много времени и

памяти. База знаний содержит 500...1000 правил. Время разработки — 6...12 мес.;

промышленная экспертная система — обеспечивает высокое качество решения всех задач при минимуме времени и памяти, что достигается переписыванием программ с использованием более совершенных инструментальных средств и языков низкого уровня. База знаний содержит 1000...1500 правил. Время разработки — 1...1,5 года;

коммерческая экспертная система — отличается от промышленной тем, что помимо собственного использования она может продаваться различным потребителям. База знаний содержит 1500...3000 правил. Время разработки — 1,5...3 года. Стоимость — 0,3...5 млн долларов.

В настоящее время уже сложилась определенная технология разработки экспертных систем, которая состоит из следующих этапов, схематично изображенных на рис. 2.5.

1. *Идентификация* (постановка задачи). На этапе устанавливаются задачи, которые подлежат решению, выявляются цели разработки, требования к экспертной системе, ресурсы, используемые понятия и их взаимосвязи, определяются методы решения задач. Цель этапа — сформулировать задачу, охарактеризовать поддерживающую ее базу знаний и таким образом обеспечить начальный импульс для развития базы знаний.

2. *Концептуализация*. Проводится содержательный анализ проблемной области, выявляются используемые понятия и их взаимосвязи, определяются методы решения задач.

3. *Формализация*. Определяются способы представления всех видов знаний, формализуются основные понятия, определяются способы интерпретации знаний, оценивается адекватность целям

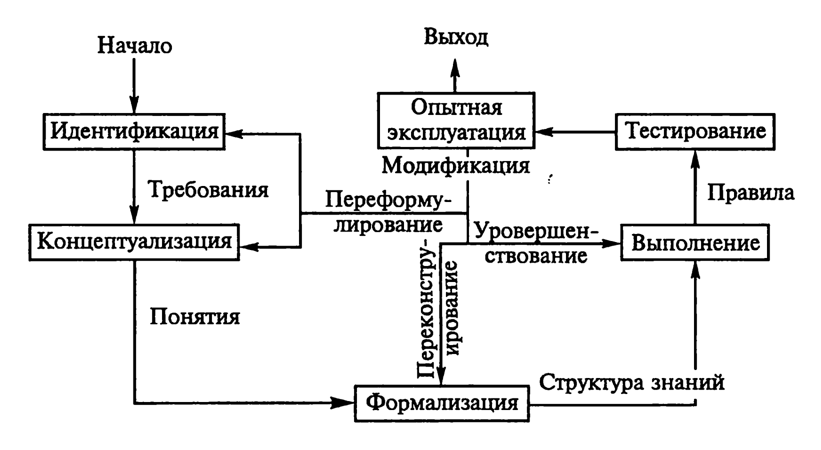


Рис. 2.5. Технология разработки экспертной системы

системы зафиксированных понятий, методов решения, средств представления и манипулирования знаниями.

4. *Выполнение.* Осуществляется наполнение экспертом базы знаний. Процесс приобретения знаний разделяют на извлечение знаний из эксперта, организацию знаний, обеспечивающую эффективную работу системы, и представление знаний в виде, понятном экспертной системе. Из-за эвристического характера знаний их приобретение является весьма трудоемким.

5. *Тестирование.* Эксперт и инженер по знаниям в интерактивном режиме, используя диалоговые и объяснительные средства, проверяют компетентность экспертной системы. Процесс тестирования продолжается до тех пор, пока эксперт не решит, что система достигла требуемого уровня компетентности.

6. *Опытная эксплуатация.* Проверяется пригодность экспертной системы для конечных пользователей. По результатам этого этапа может потребоваться модификация экспертной системы.

7. *Модификация.* В ходе создания экспертной системы почти постоянно производится ее модификация: переформулирование понятий и требований, переконструирование представления знаний и усовершенствование прототипа.

Усовершенствование прототипа осуществляется в процессе циклического прохождения через этапы выполнения и тестирования для отладки правил и процедур вывода.

Переконструирование выбранного ранее способа представления знаний предполагает возврат с этапа тестирования на этап формализации.

Если возникшие проблемы еще более серьезны, то после неудачи на этапе тестирования может потребоваться возврат на этап концептуализации и идентификации. В этом случае речь идет о переформулировании понятий, используемых в системе, т.е. перепроектировании системы заново.

Приведенная последовательность разработки экспертных систем предложена Э.А.Поповым в книге [14]. Данная технология отражает опыт разработки и внедрения многочисленных экспертных систем широкого назначения. В этой же книге указываются трудности, характерные для каждой стадии, и даются рекомендации по их преодолению.

Контрольные вопросы

1. Назовите общие и отличительные признаки данных и знаний.
2. Назовите и охарактеризуйте известные вам методы представления знаний.
3. Какой по вашему мнению метод представления знаний используется в человеческом мозге?
4. Приведите примеры формализованных и неформализованных знаний.

5. Дайте определение и сформулируйте назначение экспертной системы.

6. Приведите примеры известных вам экспертных систем.

7. Что такое оболочка экспертной системы?

8. Каким по вашему мнению должен быть коллектив разработчиков экспертной системы?

9. Перечислите и охарактеризуйте стадии и этапы разработки экспертных систем.

ГЛАВА 3

НЕЙРОИНФОРМАТИКА

3.1. ПЕРСЕПТРОН И ЕГО РАЗВИТИЕ

3.1.1. Мозг и компьютер

На самой заре компьютерной эры в середине XX в. были предложены различные варианты принципов действия и архитектурного исполнения электронно-вычислительных машин. Многие из этих вариантов не получили дальнейшего развития и были забыты. Наиболее плодотворной оказалась архитектура машины фон Неймана, которую имеет большинство современных компьютеров. Однако наряду с машиной фон Неймана до наших дней дошла еще одна схема, которая в последние годы получила стремительное развитие и применение. Речь идет о нейросетевых и нейрокомпьютерных технологиях.

Нейронные сети и нейрокомпьютеры — это одно из направлений компьютерной индустрии, в основе которого лежит идея создания искусственных интеллектуальных устройств по образу и подобию человеческого мозга. Дело в том, что компьютеры, выполненные по схеме машины фон Неймана, по своей структуре и свойствам весьма далеки от нашего естественного компьютера — человеческого мозга. В подтверждение этому в табл. 3.1. приведены признаки, отличающие человеческий мозг от неймановского компьютера.

Основатели же нейрокибернетики задались целью создания электронных устройств, структурно и функционально адекватных мозгу. Но прежде чем рассматривать такие устройства, приведем основные сведения о принципах организации и функционирования человеческого мозга.

Мозг человека состоит из белого и серого вещества: белое — это тела нейронов, а серое — соединяющие их нервные волокна. Каждый нейрон состоит из трех частей: *тела клетки*, *дендритов* и *аксона.*

Нейрон получает информацию через свои дендриты, а передает ее дальше через аксон, разветвляющийся на конце на тысячи *синапсов* — нервных нитей, соединяющих нейроны между собой (рис. 3.1). Простейший нейрон может иметь до 10 000 дендритов, принимающих сигналы от других клеток. В человеческом мозге содержится приблизительно 10^{11} нейронов. Каждый нейрон связан с $10^3 \dots 10^4$ другими нейронами. Таким образом, биологическая

нейронная сеть, составляющая мозг человека, содержит $10^{14}...10^{15}$ взаимосвязей.

Каждый нейрон может существовать в двух состояниях — возбужденном и невозбужденном. В возбужденное состояние нейрон переходит под воздействием электрических сигналов, поступающих к нему от других нейронов, когда эти воздействия становятся достаточно большими. В возбужденном состоянии нейрон сам посылает электрический сигнал другим соединенным с ним нейронам.

Нейроны взаимодействуют между собой посредством коротких серий импульсов продолжительностью несколько микросекунд. Частота импульсов составляет от нескольких единиц до сотен герц, что в миллион раз медленнее, чем в современных электронных схемах. Тем не менее, такие сложные операции, как распознавание зрительного образа, человек выполняет за несколько сотен микросекунд. Если учесть, что скорость выполнения операций нейронами составляет единицы микросекунд, то вся операция распознавания требует около 100 последовательных нейронных операций. Это значит, что при распознавании образов человеческий мозг запускает параллельные программы, каждая из которых имеет не более ста шагов. Сделанный вывод известен под названием «правило ста шагов».

Таблица 3.1

Сопоставление принципов построения и свойств современного компьютера (машины фон Неймана) и человеческого мозга

Параметр	Машина фон Неймана	Человеческий мозг
Процессор	Сложный	Простой
	Высокоскоростной	Низкоскоростной
	Один или несколько	Большое количество
Память	Отделена от процессора	Интегрирована в процессор
	Локализованная	Распределенная
	Адресация не по содержанию	Адресация по содержанию
Вычисления	Централизованные	Распределенные
	Последовательные	Параллельные
	По хранимым программам	По самообучающимся программам
Надежность	Высокая уязвимость	Живучесть
Среда фунцио-нирования	Строго определенная	Плохо определенная
	Строго ограниченная	Без ограничений

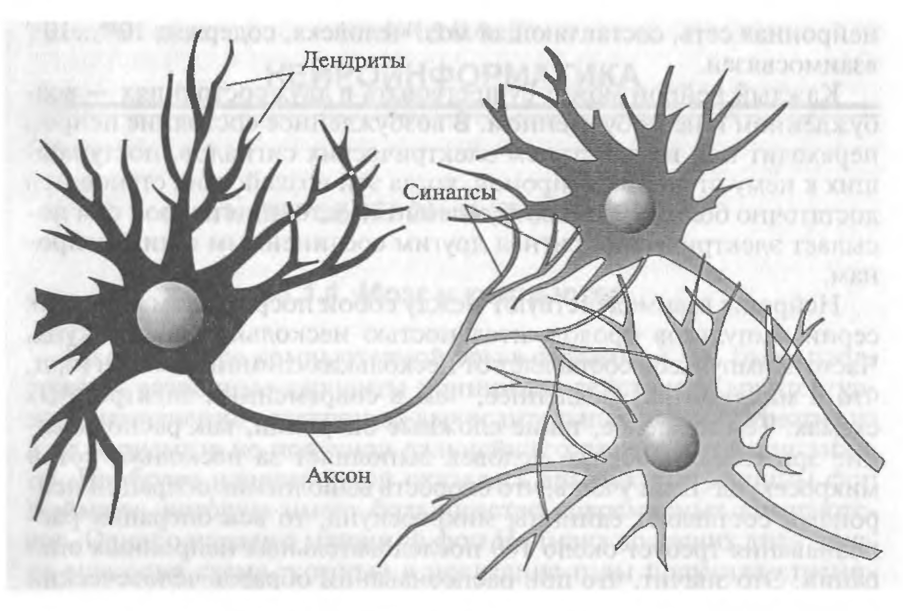


Рис. 3.1. Нейроны человеческого мозга

Известно, что общее число нейронов в течение жизни человека практически не изменяется, т.е. мозг ребенка и мозг взрослого человека содержат приблизительно одинаковое число нейронов. Примерно одинаковое число нейронов содержат мозг ученого, политического деятеля и спортсмена. Отличие состоит в *силе синаптических связей*, т.е. в величине электрических проводимостей нервных волокон, соединяющих нейроны. На этом основании была высказана гипотеза о том, что все наши мысли, эмоции, знания, вся информация, хранящаяся в человеческом мозге, закодирована в виде сил синаптических связей. Если учесть, что таких связей в человеческом мозге $10^{14}...10^{15}$, то получается, что именно такой размер имеет матрица кодов хранимой информации. Процесс же обучения человека, продолжающийся всю его жизнь, состоит в непрерывной корректировке содержимого этой матрицы.

3.1.2. Математический нейрон Мак-Каллока—Питтса

Исторически первой работой, заложившей теоретический фундамент для создания интеллектуальных устройств, не только функционально, но и структурно моделирующих человеческий мозг, принято считать опубликованную в 1943 г. статью Уоррена Мак-Каллока и Вальтера Питтса [62]. Ее авторы выдвинули гипотезу *математического нейрона* — абстрактного устройства, моделирующего нейрон мозга человека. Математический нейрон тоже име-

ет несколько входов и один выход. Через входы, число которых обозначим J, математический нейрон принимает входные сигналы x_j, которые суммирует, умножая каждый входной сигнал на некоторый весовой коэффициент w_j:

$$S = \sum_{j=1}^{J} w_j x_j. \qquad (3.1)$$

Выходной сигнал нейрона y может принимать одно из двух значений — нуль или единицу, которые формируются следующим образом:

$$y = 1, \text{ если } S \geq \theta; \qquad (3.2)$$
$$y = 0, \text{ если } S < \theta, \qquad (3.3)$$

где θ — порог чувствительности нейрона.

Таким образом, математический нейрон, как и его биологический прототип, существует в двух состояниях. Если взвешенная сумма входных сигналов S не достигает некоторой пороговой величины θ, то математический нейрон не возбужден и его выходной сигнал равен нулю. Если же входные сигналы достаточно интенсивны и их сумма достигает порога чувствительности, то нейрон переходит в возбужденное состояние и на его выходе образуется сигнал $y = 1$. Весовые коэффициенты w_j имитируют электропроводность нервных волокон — силу синаптических связей между нейронами. Чем они выше, тем больше вероятность перехода нейрона в возбужденное состояние. Логическая функция (3.2), (3.3), называемая *активационной функцией* нейрона, графически изображена на рис. 3.2.

Таким образом, математический нейрон представляет собой пороговый элемент с несколькими входами и одним выходом. Одни из входов математического нейрона оказывают возбуждающее действие, другие — тормозящее. Каждый математический нейрон имеет свое определенное значение порога. На рис. 3.3 приведены схематические представления математических нейронов, связанных между собой в нейронную сеть.

Математический нейрон обычно изображают кружочком, возбуждающий вход — стрелкой, а тормозящий — маленьким кружочком. Рядом может записываться число, показывающее значение порога θ. Как показано на рис. 3.4, математические нейроны могут реализовывать различные логические функции. Так, математический нейрон, имеющий два входа с единичными силами синаптических связей $w_1 = w_2 = 1$, согласно формулам (3.1)—(3.3) реализует функцию логического умножения «И» при $\theta = 2$

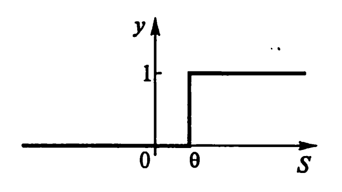


Рис. 3.2. Пороговая активационная функция нейрона

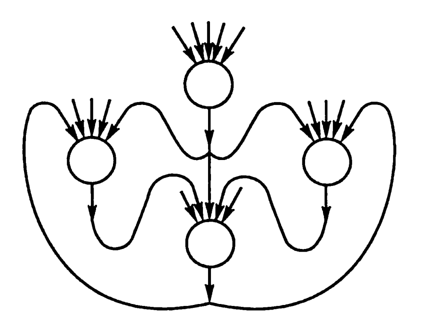


Рис. 3.3. Схематическое изображение участка нейронной сети

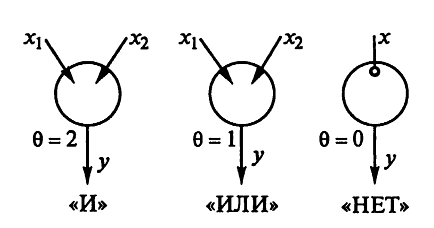


Рис. 3.4. Математические нейроны, реализующие логические функции

и функцию логического сложения «ИЛИ» при $\theta = 1$. Нейрон с одним входом, у которого $w = -1$, реализует логическую функцию «НЕТ» при $\theta = 0$.

3.1.3. Персептрон Розенблатта и правила Хебба

У. Мак-Каллок и В. Питтс предложили конструкцию сети из математических нейронов и показали, что такая сеть в принципе может выполнять числовые и логические операции. Далее они высказали идею о том, что сеть из математических нейронов в состоянии обучаться, распознавать образы, обобщать, т. е. она обладает свойствами человеческого интеллекта.

Идея Мак-Каллока—Питтса была материализована в 1958 г. Фрэнком Розенблаттом сначала в виде компьютерной программы для ЭВМ IBM-794 [63], а затем, спустя два года, в виде электронного устройства, моделирующего человеческий глаз [64]. Это устройство, имеющее в качестве элементной базы модельные нейроны Мак-Каллока—Питтса и названное персептроном, удалось обучить решению сложнейшей интеллектуальной задачи — распознаванию букв латинского алфавита. Таким образом, удалось проверить основные гипотезы функционирования человеческого мозга и сам механизм его обучаемости. «Нельзя сказать, что мы точно воспроизводим работу человеческого мозга, — признавал Ф. Розенблатт, — но пока персептрон ближе всего к истине».

Разберем принцип действия персептрона на примере решения конкретных задач. На рис. 3.5 приведен один из простейших вариантов исполнения персептрона, предназначенного для классификации цифр на четные и нечетные. Представим себе матрицу из 12 фотоэлементов, расположенных в виде четырех горизонтальных рядов, в каждом из которых три фотоэлемента. На матрицу фотоэлементов накладывается карточка с изобра-

жением цифры (на рис. 3.5 — это цифра 4). Если на фотоэлемент попадает какой-либо фрагмент цифры, то данный фотоэлемент вырабатывает сигнал в виде двоичной единицы, в противном случае — нуль. На рис. 3.5 первый фотоэлемент выдает сигнал $x_1 = 0$, второй фотоэлемент — $x_2 = 1$ и т.д. Согласно формулам (3.1)—(3.3) персептронный нейрон выполняет суммирование входных сигналов x_j, помноженных на синаптические веса w_j, первоначально заданные датчиком случайных чисел. После этого сумма сравнивается с порогом чувствительности θ, также заданным случайным образом. Цель обучения персептрона состоит в том, чтобы выходной сигнал y был равен единице, если на карточке была изображена четная цифра, и нулю, если цифра была нечетной.

Эта цель достигается путем обучения персептрона, заключающемся в корректировке весовых коэффициентов w_j. Если, например, на вход персептрона была предъявлена карточка с цифрой 4 и выходной сигнал y случайно оказался равным единице, означающей четность, то корректировать веса не нужно, так как реакция персептрона правильна. Однако если выход неправилен и $y = 0$, то следует увеличить веса тех активных входов, которые способствуют возбуждению нейрона. В данном случае увеличению подлежат w_2, w_j, w_{11} и др.

Таким образом, можно сформулировать следующий итерационный алгоритм корректировки весовых коэффициентов, обеспечивающий обучение персептрона в нужном направлении.

Шаг 1. Подать входной образ и вычислить выход персептрона y.

Шаг 2, а. Если выход правильный, то перейти на шаг 1.

Шаг 2, б. Если выход неправильный и равен нулю, то увеличить веса активных входов, например добавить все входы к соответствующим им весам: $w_j(t + 1) = w_j(t) + x_j$.

Шаг 2, в. Если выход неправильный и равен единице, то уменьшить веса активных входов, например вычесть каждый вход из соответствующего ему веса: $w_j(t + 1) = w_j(t) - x_j$.

Шаг 3. Перейти на шаг 1 или завершить процесс обучения.

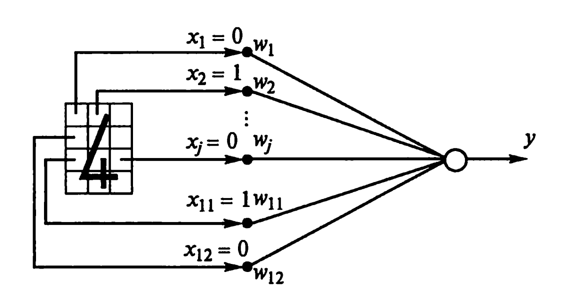


Рис. 3.5. Персептрон, классифицирующий цифры на четные и нечетные

В приведенном здесь алгоритме шаг 2, б называют *первым правилом Хебба*, а шаг 2, в — *вторым правилом Хебба*, в честь ученого, предложившего этот алгоритм в 1949 г. [55]. Отметим, что правила Хебба удивительным образом напоминают процесс обучения ребенка методом поощрения—наказания или дрессировки животного методом «кнута и пряника». Как и в случаях с ребенком и животным, алгоритм обучения персептрона за конечное число попыток (итераций, или *эпох*) может привести к цели — персептрон научится различать четные и нечетные цифры.

Возникает вопрос: «Всегда ли алгоритм обучения персептрона приводит к желаемому результату?». Ответ на этот вопрос дает теорема сходимости персептрона, формулируемая следующим образом.

Если существует множество значений весов, которые обеспечивают конкретное различение образов, то в конечном итоге алгоритм обучения персептрона приводит либо к этому множеству, либо к эквивалентному ему множеству, такому, что данное различение образов будет достигнуто.

Интересно отметить, что по числу выполненных доказательств теорема сходимости персептрона занимает одно из первых мест в мире [31]. Ранее самой доказанной в мире теоремой считалась теорема Пифагора.

3.1.4. Дельта-правило и распознавание букв

Рассмотренный алгоритм обучения персептрона можно представить в более общей форме. Если через d обозначить требуемый выходной сигнал, то на каждой итерации можно рассчитывать разницу между требуемым ответом персептрона d и реальным значением y, вычисляемым на его выходе:

$$\varepsilon = (d - y). \tag{3.4}$$

Тогда случай $\varepsilon = 0$ соответствует шагу 2, а, когда выход правилен; случай $\varepsilon > 0$ — шагу 2, б; случай $\varepsilon < 0$ — шагу 2, в.

Идея алгоритма обучения персептрона с помощью правил Хебба сохранится, если итерационный процесс вести по формулам:

$$w_j(t+1) = w_j(t) + \Delta w_j; \tag{3.5}$$

$$\Delta w_j = \varepsilon x_j, \tag{3.6}$$

где $w_j(t)$ и $w_j(t+1)$ — соответственно старое и новое значения весовых коэффициентов персептрона; j — номер входного сигнала.

Кроме того, можно получить аналогичную итерационную формулу для подстройки порогового значения нейрона θ, если учесть, что его можно интерпретировать как вес дополнительного входа x_0, значение которого равно -1:

$$\theta(t+1) = \theta(t) + \Delta\theta; \tag{3.7}$$

$$\Delta\theta = -\varepsilon. \tag{3.8}$$

В итерационные формулы можно ввести коэффициент скорости обучения η, с помощью которого можно управлять величиной коррекции весов:

$$\Delta w_j = \eta\varepsilon x_j; \tag{3.9}$$

$$\Delta\theta = -\eta\varepsilon. \tag{3.10}$$

Алгоритм обучения персептрона с использованием этих формул известен под названием *дельта-правила*. Дальнейшее развитие идеи персептрона и алгоритмов обучения связано с усложнением его структуры и развитием функциональных свойств.

На рис. 3.6 приведена схема персептрона, предназначенного для распознавания букв русского алфавита. В отличие от предыдущей схемы такой персептрон имеет 33 нейрона, таким образом, каждой букве алфавита соответствует свой нейрон. Полагается, что выход первого нейрона y_1 должен быть равен единице, если персептрону предъявлена буква «А», и нулю для всех остальных букв. Выход второго нейрона y_2 должен быть равен единице, если персептрону предъявлена буква «Б», и нулю во всех остальных случаях. И так далее до буквы «Я».

Алгоритм обучения данного персептрона выглядит следующим образом.

Шаг 1. Датчиком случайных чисел всем весовым коэффициентам w_{ij} и пороговым значениям нейронов θ_i ($i = 1, ..., 33$, $j = 1, ..., 12$) присваиваются некоторые малые значения.

Шаг 2. Персептрону предъявляется какая-либо буква алфавита, и системой фотоэлементов вырабатывается входной вектор x_j ($j = 1, ..., 12$).

Шаг 3. Каждый нейрон выполняет взвешенное суммирование входных сигналов

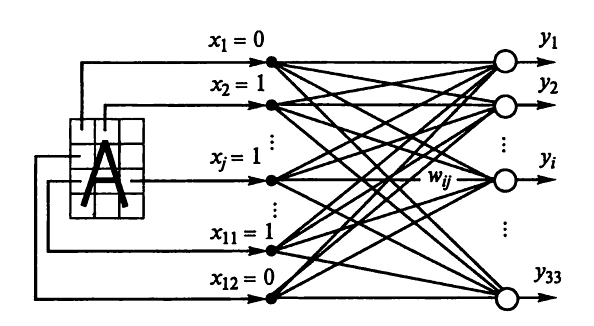


Рис. 3.6. Персептрон, предназначенный для распознавания букв русского алфавита

$$S_i = \sum_{j=1}^{12} w_{ij} x_j$$

и вырабатывает выходной сигнал $y_i = 1$, если $S_i \geq \theta_i$; $y_i = 0$, если $S_i < \theta_i$.

Шаг 4. Для каждого нейрона вычисляется ошибка

$$\varepsilon_i = (d_i - y_i),$$

где d_i — вектор правильных ответов персептрона (например, для буквы «А» $d_1 = 1$, $d_2 = 0$, ..., $d_{33} = 0$ и т.д.).

Шаг 5. Производится корректировка весовых коэффициентов персептрона и пороговых значений нейронов:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}; \quad \Delta w_{ij} = \eta \varepsilon_i x_j;$$

$$\theta_i(t+1) = \theta_i(t) + \Delta\theta_i; \quad \Delta\theta_i = -\eta \varepsilon_i,$$

где t — номер итерации, или *эпохи*.

Шаг 6. Повторение шагов 2—5 необходимое число раз.

3.1.5. Адалайн, мадалайн и обобщенное дельта-правило

Персептрон, схема которого приведена на рис. 3.6, предназначен для распознавания букв алфавита. Однако его можно попытаться использовать и для решения других практических задач. Например, обучить выполнять прогноз погоды или ставить диагнозы болезней. Все зависит от того, какой смысл придавать входному вектору x_j и выходному вектору y_i. Круг решаемых задач значительно расширится, если научить персептрон выдавать не только бинарные выходные сигналы типа ноль и единица, но и аналоговые, т.е. имеющие непрерывные значения. Такое обобщение персептрона было сделано Уидроу и Хоффом [68], которые вместо ступенчатой (см. рис. 3.2) ввели непрерывную нелинейную функцию активации

$$y = \frac{1}{1 + \mathrm{e}^{-S}}, \tag{3.11}$$

график которой изображен на рис. 3.7. Эту функцию назвали *сигмоидой* из-за того, что ее графическое изображение напоминает латинскую букву «S». Другое название сигмоиды — *логистическая* функция.

Подобно обычной пороговой функции активации, сигмоида отображает точки области определения $(-\infty, +\infty)$ на интервал $(0, +1)$. Практически сигмоида обеспечивает непрерывную аппроксимацию классической пороговой функции. Для сигмоиды принято обозначение $y = f_\sigma(S)$.

Персептроны с сигмоидными активационными функциями с одним выходом назвали *адалайн*, с несколькими выходами — *ма-*

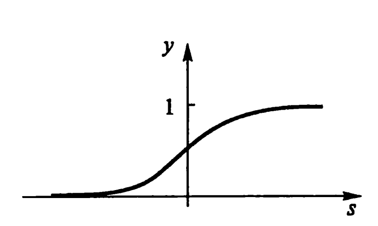


Рис. 3.7. Сигмоидная активационная функция $y = f_\sigma(S)$

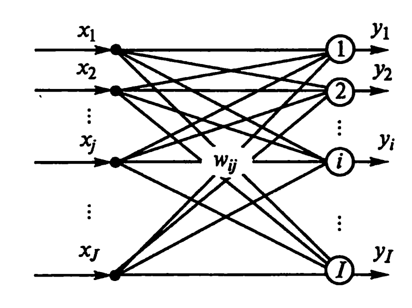


Рис. 3.8. Однослойный персептрон с J входами и I выходами

далайн (от английских слов ADAptive LInear NEuron и Many ADALINE). Появление персептронов с непрерывными активационными функциями обусловило и новые подходы к их обучению. Уидроу и Хофф [68, 69] предложили минимизировать среднеквадратичную ошибку, определяемую как разность между требуемыми d_i и реальными y_i выходными сигналами персептрона:

$$\varepsilon = \frac{1}{2}\sum_{i=1}^{I}(d_i - y_i)^2. \qquad (3.12)$$

Рассмотрим алгоритм коррекции весовых коэффициентов персептрона, имеющего J входов и I выходов (рис. 3.8). Среднеквадратичная ошибка ε является многомерной функцией весовых коэффициентов, т.е. $\varepsilon = \varepsilon(w_{ij})$, и в пространстве координат w_{ij} представляется в виде некоторой многомерной поверхности — *гиперповерхности*. Если оставить только две оси координат, например w_{11} и w_{12}, то эта поверхность будет иметь вид фигуры, напоминающей параболоид (рис. 3.9), который, однако, может иметь как один, так и несколько минимумов. Поэтому такую поверхность будем называть *псевдопараболоидом*. Обучение персептрона можно представить как задачу отыскания такого сочетания весовых коэффициентов w_{ij}, которому соответствует самая нижняя точка *гиперпсевдопараболоида*. Такую задачу называют *оптимизационной* и говорят, что она состоит в минимизации функционала $\varepsilon = \varepsilon(w_{ij})$ в пространстве параметров w_{ij}.

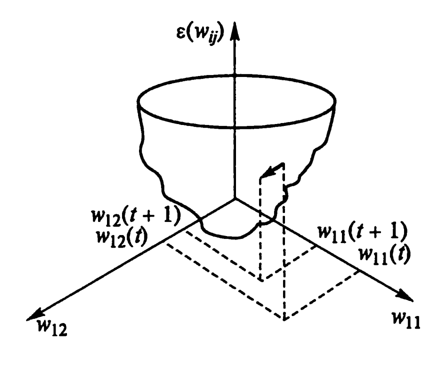


Рис. 3.9. Псевдопараболоид, изображающий зависимость среднеквадратичной ошибки ε от весовых коэффициентов w_{11} и w_{12}

Существует множество методов решения оптимизационных задач. Наиболее простым являет-

ся случайный перебор весовых коэффициентов w_{ij} с последующими вычислениями и сравнениями между собой соответствующих этим коэффициентам функций ошибок ε. Однако более эффективным является метод *градиентного спуска*, согласно которому изменение (коррекция) каждого весового коэффициента Δw_{ij} производится в сторону, противоположную градиенту поверхности гиперпсевдопараболоида, т.е.

$$\Delta w_{ij} = -\eta \frac{\partial \varepsilon}{\partial w_{ij}}, \tag{3.13}$$

где η — коэффициент скорости обучения.

Среднеквадратичная ошибка ε является сложной функцией, зависящей, в первую очередь, от выходных сигналов персептрона y_i, поэтому

$$\frac{\partial \varepsilon}{\partial w_{ij}} = \frac{\partial \varepsilon}{\partial y_i} \frac{\partial y_i}{\partial w_{ij}}. \tag{3.14}$$

Здесь $y_i = f_\sigma(S_i)$, где $S_i = \sum_{j=1}^{J} w_{ij} x_j$. Следовательно,

$$\frac{\partial y_i}{\partial w_{ij}} = \frac{\partial f_\sigma(S_i)}{\partial S_i} \frac{\partial S_i}{\partial w_{ij}} = f'_\sigma(S_i) x_j. \tag{3.15}$$

Кроме того, если продифференцировать (3.12) по y_n, где $n \in [1, I]$, то получится $\frac{\partial \varepsilon}{\partial y_n} = -(d_n - y_n)$, значит,

$$\frac{\partial \varepsilon}{\partial y_i} = -(d_i - y_i). \tag{3.16}$$

Подставив (3.15) и (3.16) в (3.14) и затем полученное выражение в (3.13), окончательно будем иметь

$$\Delta w_{ij} = -\eta\big(-(d_i - y_i) f'_\sigma(S_i) x_j\big) = \eta(d_i - y_i) f'_\sigma(S_i) x_j. \tag{3.17}$$

Это выражение получено для нейронов с активационными функциями любого вида. Однако, если $f_\sigma(S_i)$ — сигмоида, заданная формулой (3.11), то

$$f'_\sigma(S_i) = \left((1 + e^{-S_i})^{-1}\right)' = f_\sigma(S_i)\big(1 - f(S_i)\big). \tag{3.18}$$

Подставив это выражение в (3.17), получим

$$\Delta w_{ij} = \eta(d_i - y_i) f_\sigma(S_i)\big(1 - f(S_i)\big) x_j = \eta(d_i - y_i) y_i (1 - y_i) x_j. \tag{3.19}$$

Итак, мы получили итерационную формулу для обучения однослойного персептрона

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}, \quad (3.20)$$

в которой

$$\Delta w_{ij} = \eta \delta_i x_j; \quad (3.21)$$

$$\delta_i = (d_i - y_i) y_i (1 - y_i). \quad (3.22)$$

Этот алгоритм называют обобщенным дельта-правилом, преимущество которого состоит в более быстрой сходимости и возможности более точной обработки входных и выходных непрерывных сигналов, т.е. в расширении круга решаемых персептронами задач и возможности получения более точных и качественных решений.

3.1.6. Ограниченность однослойного персептрона

Как уже отмечалось ранее, Ф. Розенблатту [63, 64] удалось обучить свой персептрон распознавать буквы алфавита. Это был колоссальный успех. Электронное устройство, созданное по образу и подобию человеческого мозга, обученное подобно человеку, успешно моделировало интеллектуальные функции человека. Это был шаг вперед в познании самой природы человеческого мышления. Мозг начал раскрывать свои тайны. Появилась возможность исследовать мозг методами моделирования, не прибегая к сложнейшим антигуманным и мало что дающим натурным экспериментам. Это была сенсация, приковавшая к себе внимание мыслящих людей всего мира. Казалось, что ключ к интеллекту был найден и полное воспроизведение человеческого мозга и всех его функций — всего лишь вопрос времени. Ученым, инженерам, бизнесменам, политикам виделись самые радужные перспективы практического применения систем искусственного интеллекта. Правительство США выделило крупные субсидии на развитие нового перспективного научного направления.

Между тем, класс решаемых персептронами задач расширялся. Делались попытки применения персептронов в задачах прогнозирования, таких как предсказание погоды и курсов акций. Персептроны применялись для решения задач диагностики, таких как анализ электрокардиограмм и заключение врача о диагнозе болезни пациента. По мере расширения фронта научных исследований появились трудности. Неожиданно оказалось, что многие новые задачи персептрон решить не мог, причем эти задачи внешне ничем не отличались от тех, с которыми персептрон успешно справлялся ранее. Возникла необходимость объяснения возникших парадоксов, глубокого анализа и создания теоретической базы персептронов.

Следующий период истории персептронов начался с появления книги М. Минского и С. Пайперта «Персептроны» [27]. В этой

книге математически строго было доказано, что использовавшиеся в то время однослойные персептроны в принципе не способны решать многие простые задачи. Одну из таких задач, заключающуюся в реализации логической операции «Исключающее ИЛИ», мы рассмотрим подробно.

«Исключающее ИЛИ» — это булева функция двух аргументов, каждый из которых может иметь значение «истинно» либо «ложно». Сама она принимает значение «истинно», когда только один из аргументов имеет значение «истинно». Во всех остальных случаях функция принимает значение «ложно»:

$$y = (x_1 \text{ AND NOT } x_2) \text{ OR} (x_2 \text{ AND NOT } x_1). \qquad (3.23)$$

Задача состоит в том, чтобы реализовать функцию (3.23) с помощью однонейронного персептрона с двумя входами x_1 и x_2 и одним выходом y (рис. 3.10).

Закодировав значение «истинно» единицей, а «ложно» — нулем, все возможные комбинации входных сигналов можно представить на плоскости x_1—x_2 в виде четырех точек: A_1, A_2, B_1, B_2, как показано на рис. 3.11. Например, точке A_1 соответствуют входные сигналы $x_1 = 0$ и $x_2 = 0$, а точке A_2 соответствуют входные сигналы $x_1 = 1$ и $x_2 = 1$. Требуемое формулой (3.23) соответствие между входными и выходными сигналами персептрона сведено в табл. 3.2, называемую таблицей истинности логической функции.

Согласно формулам (3.1)—(3.3) однонейронный персептрон, изображенный на рис. 3.10, осуществляет преобразование:

$$S = w_1 x_1 + w_2 x_2; \qquad (3.24)$$

$$y = 1, \text{ если } S \geq \theta; \qquad (3.25)$$

$$y = 0, \text{ если } S < \theta. \qquad (3.26)$$

Заменим в уравнении (3.24) S на θ:

$$w_1 x_1 + w_2 x_2 = \theta. \qquad (3.27)$$

Если в этом уравнении величины x_1 и x_2 считать переменными, а θ, w_1 и w_2 — константами, то на плоскости x_1—x_2 рассматриваемое уравнение изобразится в виде прямой линии, положение и

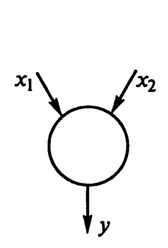


Рис. 3.10. Однонейронный персептрон с двумя входами и одним выходом

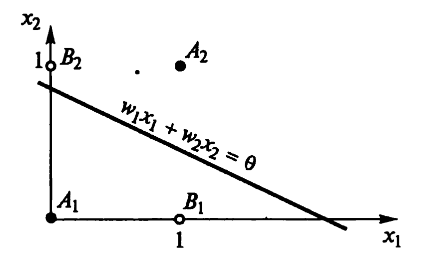


Рис. 3.11. К пояснению проблемы «Исключающего ИЛИ»

Таблица 3.2

Таблица истинности функции «Исключающее ИЛИ»

Точки	Значение x_1	Значение x_2	Значение y
A_1	0	0	0
A_2	1	1	0
B_1	1	0	1
B_2	0	1	1

наклон которой определяются значениями весовых коэффициентов w_1, w_2 и порога θ. Для всех точек плоскости $x_1—x_2$, лежащих на этой линии, выполняется равенство $S = \theta$, и поэтому, вследствие (3.25), выход персептрона равен единице. Для точек, лежащих выше указанной линии, выход персептрона также равен единице, а для точек, лежащих ниже этой линии, выход персептрона равен нулю. Поэтому линию, изображающую уравнение (3.27), называют *пороговой прямой.*

Согласно табл. 3.2 в точках A_1 и A_2 выход персептрона должен быть нулевым, а в точках B_1 и B_2 — единичным. Но для этого надо расположить пороговую прямую так, чтобы точки A_1 и A_2 лежали ниже этой линии, а точки B_1 и B_2 — выше, что невозможно. Это значит, что, какие бы значения ни придавались весам и порогу, рассмотренный персептрон в принципе не способен воспроизвести соотношение между входами и выходом, требуемое для представления функции «Исключающее ИЛИ».

Помимо проблемы «Исключающее ИЛИ» в книге [27] приведен ряд других задач, в которых входы персептронов не могут быть разделены пороговой прямой (в многомерных случаях — плоскостью, гиперплоскостью). Такие задачи получили название *линейно неразделимых.*

3.1.7. Многослойный персептрон и алгоритм обратного распространения ошибки

Появление книги «Персептроны» [27] вызвало шок в научном мире. Строгие математические доказательства М. Минского и С. Пайперта были неуязвимы. Всеобщий энтузиазм сменился не менее всеобщим пессимизмом. Правительство США прекратило финансирование нейропроектов, и персептроны были преданы забвению, длившемуся более 20 лет.

Тем не менее, работы в области нейросетевых и нейрокомпьютерных технологий продолжались отдельными наиболее настойчивыми исследователями. Многие понимали, что надо усложнять структуру персептронов, т. е. продолжать приближать компьютер-

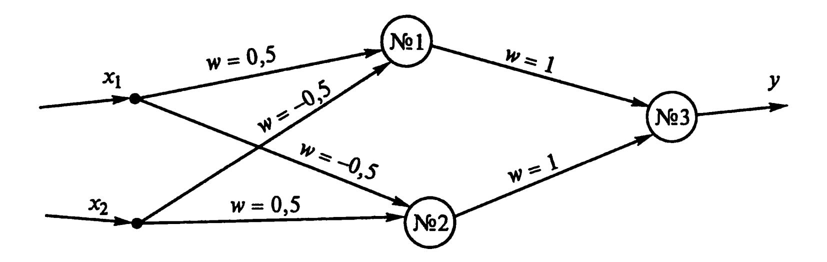


Рис. 3.12. Двухслойный персептрон, реализующий функцию «Исключающее ИЛИ»

ную модель к оригиналу — человеческому мозгу. Оказалось, что проблему «Исключающее ИЛИ» можно решить с помощью двухслойного персептрона, изображенного на рис. 3.12.

Работа этого персептрона происходит по следующему алгоритму.

Нейрон № 1:

$$S_1 = 0,5 \times x_1 + (-0,5) \times x_2;$$
$$y_1 = 1, \text{ если } S_1 \geq \theta;$$
$$y_1 = 0, \text{ если } S_1 < \theta.$$

Нейрон № 2:

$$S_2 = (-0,5) \times x_1 + 0,5 \times x_2;$$
$$y_2 = 1, \text{ если } S_2 \geq \theta;$$
$$y_2 = 0, \text{ если } S_2 < \theta.$$

Нейрон № 3:

$$S_3 = 1 \times y_1 + 1 \times y_2;$$
$$y_3 = 1, \text{ если } S_3 \geq \theta;$$
$$y_3 = 0, \text{ если } S_3 < \theta.$$

С помощью этих формул легко проверить таблицу истинности персептрона, составленную при задании порога $\theta = 0,5$ (табл. 3.3).

Таблица 3.3

Таблица истинности двухслойного персептрона (см. рис. 3.12)

x_1	x_2	S_1	S_2	y_1	y_2	S_3	y_3	y
0	0	0	0	0	0	0	0	0
0	1	–0,5	0,5	0	1	1	1	1
1	0	0,5	–0,5	1	0	1	1	1
1	1	0	0	0	0	0	0	0

Советским ученым С.О.Мкртчяном [28] был разработан специальный математический аппарат, позволяющий без обучения строить многослойные персептроны, моделирующие любые булевы функции.

Многие исследователи понимали, что объединение нейронов в нейронные сети расширяет класс задач, решаемых персептроном, но не представляли, как такие сети обучать. Простые и изящные правила Хебба и их обобщение — дельта-правило — годились только для корректировки синаптических весов нейронов выходного слоя, тогда как вопрос о настройке параметров внутренних нейронных слоев оставался открытым.

Эффективный алгоритм обучения многослойных персептронов, открывший путь для их широкого практического применения, стал известен только в 1986 г., благодаря работе Румельхарта, Хилтона и Вильямса [65]. Интересно, что данный фундаментальный алгоритм, называемый *алгоритмом обратного распространения ошибки* (*back propagation*), был предложен на один год ранее в работах Паркера и Ле-Кана, изданных независимо одна от другой. Более того, еще в 1974 г. этот простой и изящный алгоритм обратного распространения ошибки был защищен Вербосом в его докторской диссертации. Однако тогда он остался незамеченным и только спустя более десяти лет был «переоткрыт» заново и получил всеобщее признание и применение.

Рассмотрим идею алгоритма обратного распространения ошибки, попытавшись обобщить дельта-правило для случая обучения двухслойного персептрона, имеющего N входов, I выходов и скрытый слой из J нейронов (рис. 3.13). Алгоритм корректировки синаптических весов нейронов выходного слоя оставим таким же, как для однослойного персептрона (см. формулы (3.20)—(3.22), заменив x_j на y_j:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}; \qquad (3.28)$$

$$\Delta w_{ij} = \eta \delta_i y_j; \qquad (3.29)$$

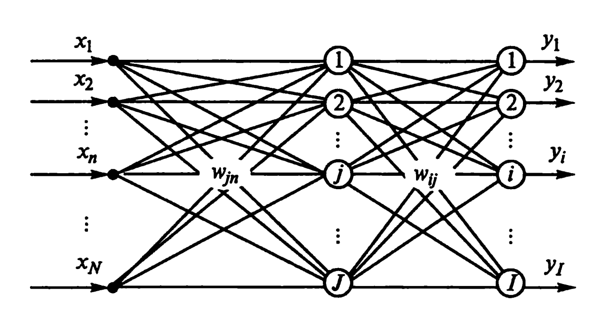


Рис. 3.13. Двухслойный персептрон с N входами, I выходами и скрытым слоем из J нейронов

$$\delta_i = (d_i - y_i)y_i(1 - y_i). \qquad (3.30)$$

Синаптические веса нейронов скрытого слоя попытаемся корректировать с помощью тех же самых формул, в которых индекс i заменим на j, а индекс j — на n:

$$\Delta w_{jn} = \eta \delta_j y_n; \qquad (3.31)$$

$$\delta_j = (d_j - y_j)y_j(1 - y_j). \qquad (3.32)$$

Понятно, что в последнем выражении в качестве y_n следует использовать x_n. Однако неясным здесь остается вопрос о вычислении нейронной ошибки ($d_j - y_j$), которая для скрытого слоя неизвестна. Идея авторов рассматриваемого алгоритма состояла в том, чтобы в качестве этой ошибки использовать суммарные ошибки с выходного слоя, помноженные на силы соответствующих синаптических связей, т. е.

$$(d_j - y_j) = \sum_{i=1}^{I} \delta_i w_{ij}. \qquad (3.33)$$

Итак, для скрытого слоя окончательно имеем:

$$\Delta w_{jn} = \eta \delta_j x_n; \qquad (3.34)$$

$$\delta_j = y_j(1 - y_j)\sum_{i=1}^{I} \delta_i w_{ij}. \qquad (3.35)$$

Используя эту идею, несложно расписать алгоритм обратного распространения ошибки для обучения персептрона, имеющего произвольное число скрытых слоев. Однако прежде внесем еще одно изменение в модель нейрона. К сумме, которую вычисляет нейрон, полезно добавить некоторое число, называемое *смещением* (порог, взятый со знаком минус):

$$S_i = \sum_{j=1}^{J} x_j w_{ij} + w_{i0}. \qquad (3.36)$$

Смещение w_{i0} задается так же, как и синаптические веса w_{ij}, т. е. датчиком случайных чисел. Ввод смещения в формулу преобразования нейрона можно интерпретировать как добавление еще одного входного сигнала x_0, который всегда равен единице. Поэтому, чтобы не усложнять выкладки, сумму (3.36) представим в более компактном виде:

$$S_i = \sum_{j=0}^{J} x_j w_{ij}, \qquad (3.37)$$

приняв $x_0 = 1$.

Алгоритм обратного распространения ошибки распишем для многослойного персептрона, имеющего входной слой $k = 0$, несколько

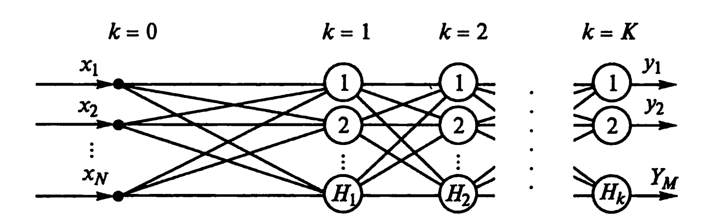


Рис. 3.14. Многослойный персептрон (MLP — MultiLayerPerseptron)

скрытых слоев $k = 1, 2, ..., K - 1$ и выходной слой $k = K$ (рис. 3.14). Нейроны входного слоя математических преобразований не выполняют, а лишь передают входные сигналы нейронам первого слоя. Будем полагать, что каждый k-й слой содержит H_k нейронов. Таким образом, персептрон имеет $N = H_0$ входов и $M = H_K$ выходов. В алгоритме будем использовать следующие обозначения: i — порядковый номер нейрона k-го слоя; j — порядковый номер нейрона $(k - 1)$-го слоя; l — порядковый номер нейрона $(k + 1)$-го слоя.

Шаг 1. Инициализация синаптических весов и смещений.

В циклах по $k = 1, 2, ..., K$; $i = 1, 2, ..., H_k$; $j = 1, 2, ..., H_{k-1}$ синаптическим весам и смещениям $w_{ij}^{(k)}$ датчиком случайных чисел присваиваются малые величины, например, из интервала от −1 до 1.

Шаг 2. Представление из обучающей выборки очередного входного вектора $\boldsymbol{X}_q = (x_1, x_2, ..., x_N)_q$ и соответствующего ему желаемого выходного вектора $\boldsymbol{D}_q = (d_1, d_2, ..., d_M)_q$, где q — номер примера в обучающей выборке.

Шаг 3. Прямой проход.

В циклах по $k = 1, 2, ..., K$; $i = 1, 2, ..., H_k$ вычисляются выходные сигналы i-го нейрона в k-м слое

$$y_i^{(k)} = f_\sigma\left(\sum_{j=0}^{H_{k-1}} w_{ij}^{(k)} y_j^{(k-1)}\right), \tag{3.38}$$

где $y_j^{(0)} = x_j$, $x_0 = 1$, $y_0^{(k-1)} = 1$, и выходные сигналы персептрона

$y_i = y_i^{(K)}$.

Шаг 4. Обратный проход.

В циклах по $k = K, K - 1, ..., 1$; $i = 1, 2, ..., H_k$; $j = 1, 2, ..., H_{k-1}$ вычисляются синаптические веса на новой эпохе

$$w_{ij}^{(k)}(t+1) = w_{ij}^{(k)}(t) + \Delta w_{ij}^{(k)}(t+1), \tag{3.39}$$

где

$$\Delta w_{ij}^{(k)}(t+1) = \eta \delta_i^{(k)} y_j^{(k-1)}, \tag{3.40}$$

причем для выходного слоя $k = K$ согласно (3.30)

$$\delta_i^{(K)} = (d_i - y_i)y_i(1 - y_i),$$

а для всех других случаев согласно (3.35)

$$\delta_i^{(k)} = y_i^{(k)}(1 - y_i^{(k)}) \sum_{l=1}^{H_{k+1}} \delta_l^{(k+1)} w_{li}^{(k+1)}. \qquad (3.41)$$

Шаг 5. Повторение шагов 2—4 необходимое число раз.

Входные векторы обучающих примеров $\boldsymbol{X}_q$ и $\boldsymbol{D}_q$ на втором шаге алгоритма обычно представляются последовательно от первого до последнего, т.е. $q = 1, 2, ..., Q$, где Q — общее число примеров. Например, в случае распознавания букв русского алфавита $Q = 33$. После того как для каждого обучающего примера будут скорректированы весовые коэффициенты персептрона, т.е. шаги 2—4 будут повторены 33 раза, на пятом шаге алгоритма вычисляется среднеквадратичная ошибка, усредненная по всем обучающим примерам:

$$\varepsilon = \frac{1}{Q} \sum_{q=1}^{Q} \sum_{i=1}^{M} (d_i - y_i)^2. \qquad (3.42)$$

Помимо среднеквадратичной ошибки может быть также оценена максимальная разность между желаемым и фактическим выходами персептрона:

$$\varepsilon = \max\left(\left|d_i - y_i\right|\right)_q, \quad i = 1, 2, ..., M, \quad q = 1, 2, ..., Q. \qquad (3.43)$$

Итерационный процесс заканчивается после того, как погрешность ε, вычисляемая по формулам (3.42) или (3.43), достигнет заданной величины, либо при достижении предельного числа эпох обучения. В результате персептрон обучится выполнять нужное отображение любого входного вектора $\boldsymbol{X}_q$ на выходной вектор $\boldsymbol{Y}_q$, отличающийся от желаемого вектора $\boldsymbol{D}_q$ на некоторую малую величину.

Теперь представим себе, что на входное табло фотоэлементов попала карточка с какой-либо буквой, выполненной другим шрифтом. Фотоэлементы сформируют входной вектор $\boldsymbol{X}$, не совпадающий ни с одним из векторов из использованной обучающей выборки. Если шрифт, которым выполнена входная буква не слишком отличается от шрифта обучающей выборки, а персептрон хорошо спроектирован и обучен, то он вычислит вектор $\boldsymbol{Y}$, в котором выход нейрона, соответствующего представленной на вход букве, будет иметь максимальное значение. Таким образом, персептрон, несмотря на помехи и искажения входного образа, выдаст правильное заключение о его принадлежности к тому или иному классу. Свойство персептрона правильно реагировать на входные образы, которых не было в обучающей выборке, называется свойством *обобщения*.

3.2. ВОЗМОЖНОСТИ И ОБЛАСТИ ПРИМЕНЕНИЯ ПЕРСЕПТРОНОВ

3.2.1. Новый подход к методу математического моделирования

С появлением алгоритма обратного распространения ошибки начался период широкого практического применения нейросетевых технологий для решения самых разнообразных задач. С помощью многослойного персептрона стало возможным строить математические модели, выполняющие сложные многомерные отображения входного вектора параметров $\boldsymbol{X}$ на выходной вектор $\boldsymbol{Y}$.

Задачи подобного рода часто встречаются в самых разнообразных, казалось бы не имеющих ничего общего областях, таких как промышленность, экономика, бизнес, финансы, политология, социология, криминалистика, медицина и т.д. Практически в каждой проблеме, решаемой прикладными науками, требуется построить модель явления, процесса, объекта, т.е. выявить и математически описать зависимость одного комплекса параметров от другого, построить математические функции, которые можно использовать для более глубокого анализа объекта, например, найти оптимальное сочетание управляющих параметров, обеспечивающих максимум целевой функции, выполнить прогнозирование, предсказать, как будут развиваться события в зависимости от того или иного воздействия.

Традиционно математические модели строились путем изучения и использования фундаментальных законов природы. В результате рассмотрения этих моделей вытекали математические формулы либо формулировки краевых задач для дифференциальных уравнений.

Персептроны открыли иной подход к самой методике построения математических моделей. Появилась возможность, не задумываясь над законами физики, химии, биологии, медицины, общественного развития, исходя из одного только эмпирического опыта (обучающей выборки) строить математические модели, которые сами извлекают эти законы и позволяют их эффективно использовать для решения широкого круга практических задач.

Появился новый инструмент извлечения знаний из данных, позволяющий заново открывать фундаментальные законы природы, выявлять ранее неизвестные и никогда не исследованные зависимости и закономерности и активно использовать их для решения конкретных практических задач.

3.2.2. Диагностика в медицине

В средствах информации имеются сообщения об удачном опыте применения нейросетей для медицинской диагностики. Рассмотрим, как строятся и обучаются такие сети.

Проведем наблюдение за тем, как врач ставит диагноз болезни пациента. Прежде всего он выясняет и записывает имя, возраст, пол, место работы, затем, как правило, измеряет давление, проводит внешний осмотр, выслушивает жалобы больного, знакомится с историей его болезни, результатами анализов, изучает электрокардиограмму. В результате у врача накапливается от 20 до 100 и более параметров, характеризующих пациента и его состояние здоровья. Это и есть исходные параметры, обработав которые с помощью своих медицинских знаний и опыта, врач делает заключение о заболевании пациента — ставит диагноз его болезни.

Задавшись целью смоделировать деятельность врача с помощью персептрона, мы прежде всего должны определиться с входным вектором X и выходным вектором Y, задав их размерности, и условиться о содержимом каждого компонента. В векторе X логично предусмотреть параметры, которые врач выясняет у больного. Например, в качестве компоненты x_1 можно задать дату рождения, в качестве x_2 — закодировать пол (например, нулем или единицей), в качестве x_3 — вес больного, x_4 — артериальное давление, x_5 — температуру тела и т. д. Нелишне учесть также цвет глаз, цвет волос, знак зодиака и другие данные, определяющие особенности организма и, следовательно, влияющие на вероятность возникновения тех или иных заболеваний. В выходном векторе Y следует закодировать все возможные диагнозы болезней, которые способен обнаружить врач.

Естественно, что размерность вектора Y можно существенно снизить, если моделировать врача, специализирующегося в узкой области медицины. Так, если мы выбрали врача-кардиолога, то в векторе Y следует кодировать только кардиологические заболевания. Например, можно принять $y_1 = 1$, если у больного инфаркт, и $y_1 = 0$, если инфаркта нет. Аналогично, в качестве y_2 можно закодировать наличие или отсутствие порока сердца, y_3 — ишемической болезни сердца и т. д. Таким образом, выходной вектор персептрона Y будет состоять из множества нулей и одной или нескольких единиц (если болезней несколько). Однако диагнозы болезней лучше кодировать по пяти-, десяти- или стобалльной шкале. Тогда на этапе подготовки обучающей выборки с помощью баллов можно будет учитывать степень уверенности врача в правильности его диагноза, а на этапе эксплуатации — вероятность правильного ответа персептрона.

Далее следует подготовить набор обучающих примеров (обучающую выборку). Мы воздержимся от советов по организации со-

вместного труда эксперта-врача и программиста, в результате которого будет создано необходимое число обучающих примеров. Отметим только, что если к работе привлечь еще и паталогоанатома, исключающего ошибки диагностики врача, то появятся все основания надеяться, что обученный таким способом персептрон по качеству выставляемых диагнозов превзойдет самого врача-эксперта.

Итак, в результате длительной работы коллектива специалистов-медиков и программистов будет накоплена выборка обучающих примеров, состоящая из множества пар векторов $\boldsymbol{X}_q$ и $\boldsymbol{D}_q$ ($q = 1, 2, ..., Q$). Теперь задача состоит в том, чтобы спроектировать персептрон и путем обучения передать ему знания и опыт, содержащиеся в выборке обучающих примеров. Вопросы проектирования персептронов, т. е. подбора числа скрытых слоев, содержащихся в них нейронов и типов активационных функций, освещены в подразд. 3.3. В качестве метода обучения персептрона можно использовать рассмотренный выше алгоритм обратного распространения ошибки.

В результате персептрон должен научиться отображать любой вектор обучающей выборки $\boldsymbol{X}_q$ на вектор $\boldsymbol{Y}$, совпадающий (либо почти совпадающий) с вектором $\boldsymbol{D}_q$. Кроме того, при появлении нового пациента, характеризующегося новым входным вектором $\boldsymbol{X}$, персептрон должен вычислить для него новый вектор $\boldsymbol{Y}$, содержащий правильный диагноз, поставленный персептроном уже без помощи врача-эксперта. Другими словами, персептрон должен уметь *обобщать* переданный ему опыт на новые точки предметной области — ставить диагнозы болезней новым, не встречавшимся ранее пациентам.

3.2.3. Диагностика неисправностей сложных технических устройств

Проблема диагностики неисправностей технических устройств считается менее сложной проблемой, чем диагностика заболеваний человека. Это утверждение относится к методу математического моделирования, основанному на законах природы — физики, химии, биологии и др. Дело в том, что в техническом устройстве, каким бы сложным оно ни было, всегда известно о функциональном назначении любого сколь угодно мелкого узла, что нельзя сказать о человеке. Естественно, что это обстоятельство является серьезным препятствием, осложняющим построение адекватных математических моделей человека традиционными способами. Нейронная же сеть сама извлекает необходимые знания из обучающих примеров, сама устанавливает неизвестные законы и раскрывает функциональные связи между элементами исследуемого

объекта. Поэтому в случае нейросетевого моделирования нет принципиальной разницы между диагностикой заболеваний человека и диагностикой неисправностей технического устройства.

Нейросетевой подход к решению задач медицинской диагностики, рассмотренный ранее, можно почти без изменений применить к проблеме диагностики неисправностей сложных технических устройств. Продемонстрируем эту идею на примере диагностики неисправностей авиационных двигателей.

Специалисты, занимающиеся этой проблемой, устанавливают датчики, измеряющие параметры работы авиадвигателей во время полетов. Файл данных полетного мониторинга обычно содержит следующие параметры: номер полета, дату полета, общую наработку двигателя, температуру и давление воздуха на входе в двигатель, температуру и давление газа за турбиной, температуру лопаток, уровень и температуру масла в маслоблоке и т.д. Число полетных параметров может достигать сотни и более, что, кстати, соизмеримо с числом параметров больного при постановке диагноза его болезни.

После выполнения определенного числа полетов (около двухсот) двигатель снимают с самолета и подвергают стендовой разборке, во время которой выявляют и устраняют его дефекты. Характерными дефектами авиадвигателей являются: трещина на сопловом агрегате, забоины, разрушение дефлектора, стружка в масле и т.д. Всего их около 30 видов.

Задача инженера-диагноста состоит в том, чтобы, используя данные мониторинга, выявить дефекты двигателя до его профилактической разборки. Традиционно эта задача решается путем применения методик, основанных на физических закономерностях: каждый дефект вызывает определенные отклонения тех или иных полетных параметров работы двигателя, поэтому, анализируя их характер изменения, можно сделать предположения о появлении дефектов, вызывающих эти изменения. Понятно, что ввиду значительных объемов информации и сложности существующих взаимосвязей между дефектами и измеренными параметрами задача анализа данных полетного мониторинга и выявления дефектов авиадвигателей является далеко не тривиальной и во многих случаях решается ненадежно и некачественно.

Рассмотрим, как эту задачу можно решать с помощью нейросетевого моделирования. Прежде всего отметим, что во входном векторе персептрона $\boldsymbol{X}$ следует предусмотреть места всем параметрам полетного мониторинга, на значения которых оказывает влияние появление выявляемых дефектов. Возможные дефекты авиадвигателя можно закодировать в выходном векторе $\boldsymbol{Y}$ с помощью традиционных нулей и единиц. Векторы желаемых выходов $\boldsymbol{D}_q$ составляются по результатам стендовых разборок двигателей. В отличие от медицинской диагностики здесь нет необходимости

вводить многобалльную шкалу диагнозов, поскольку все диагнозы в обучающей выборке имеют стопроцентную достоверность, т.е. ситуация аналогична случаю, когда диагноз болезни ставит паталогоанатом при вскрытии пациента в морге.

Кроме того, разумно предположить, что при первых вылетах нового или вновь отремонтированного самолета его двигатель полностью исправен и дефектов нет, тогда как при последних вылетах самолета двигатель уже имел те самые дефекты, которые выявились при его разборке. Поэтому из всего множества параметров полетного мониторинга ценность для обучения нейросети имеют параметры первого и последнего полетов самолета. Таким образом, для каждого двигателя, побывавшего на разборке, формируется пара обучающих векторов $\boldsymbol{X}_q$ и $\boldsymbol{D}_q$.

Если число обучающих примеров будет достаточным, то правильно спроектированный многослойный персептрон обучится надежно ставить диагнозы неисправностей авиационных двигателей, в том числе и тех, которые в обучающей выборке не участвовали. А если таким персептроном оборудовать бортовой компьютер самолета, то он будет сообщать о появлении дефекта двигателя в реальном времени, т.е. как только сложится соответствующая конфигурация вектора входных параметров — результатов измерений, снимаемых во время полета самолета. Естественно, что, прежде чем попасть в кабину пилотов, сигналы персептрона должны обрабатываться компьютером, вырабатывающим инструкции о действиях экипажа, адекватных обнаруженному дефекту двигателя.

Существует еще одно преимущество нейросетевой диагностики перед традиционными диагностическими методиками, основанными на явных знаниях. Специалисты, занимающиеся традиционными методами диагностики, пытаются учесть как можно больше взаимосвязей между значениями измеряемых полетных параметров и появлениями тех или иных дефектов двигателя. В сложных технических устройствах количество таких взаимосвязей настолько велико, а характер взаимодействий настолько сложен, что построить полную математическую модель, полностью адекватную моделируемому устройству, на современном этапе развития точных наук практически невозможно. Кроме того, в сложных технических устройствах существуют и такие взаимосвязи, о которых специалисты просто не знают. Имеются также взаимосвязи, о которых специалисты догадываются, но объяснить их физическую природу не могут, а потому в расчет не принимают. Например, известно, что дефект «стружка в масле» не влияет ни на один из измеряемых датчиками параметров работы авиадвигателя, вследствие чего не существует традиционных методик, способных выявлять этот дефект. Нейросеть же появление стружки в масле легко обнаруживает, вызывая удив-

ление специалистов. Дело тут, по-видимому, в том, что, не оказывая заметного влияния на каждый отдельно взятый параметр двигателя, появление стружки в масле все-таки влияет на общую конфигурацию входного вектора, что и вызывает соответствующую реакцию нейронной сети.

3.2.4. Нейросетевой детектор лжи

Правду ли говорит ребенок, обычно легко определить по выражению его лица, движению глаз, покраснению кожи. Со взрослым человеком значительно труднее. Если измерять давление крови, то можно выяснить, что у одних людей, говорящих неправду, оно повышается, а у других — наоборот, понижается. То же самое может происходить с пульсом.

В следственной практике МВД России [36] в настоящее время применяются полиграфы, система датчиков которых измеряет до десяти параметров, таких как пульс, артериальное давление, температура тела, частота дыхания, электросопротивление участков кожи и др. Эти параметры в реальном времени отображаются на экране монитора в виде пульсирующих кривых. Заключение о правильности ответа подследственного дается компьютерной программой, анализирующей получаемые кривые с помощью набора правил, которые обобщают исследования психологов и опыт многих наблюдений.

Ненадежность заключения, производимого таким детектором лжи, обусловлена тем, что к разным людям, по-разному реагирующим на стрессовые ситуации, применяется одна и та же система решающих правил. Поэтому помимо компьютерной программы полиграфологи вынуждены применять систему дополнительных, весьма трудоемких и кропотливых приемов.

Нейросетевые технологии позволяют по-новому подойти к проблеме построения детектора лжи. Они дают возможность создать компьютерную программу, которая настраивается на каждого конкретного человека и учитывает индивидуальные особенности его организма.

Принципиально задача выявления признаков лжи с помощью нейросетевых технологий ничем не отличается от задач диагностики, рассмотренных в двух предыдущих подразделах. В качестве входного вектора персептрона $\boldsymbol{X}$ можно использовать тот же набор параметров, что и в стандартном полиграфе. Выходной вектор $\boldsymbol{Y}$ целесообразно принять состоящим всего из двух параметров: $y_1 = 1$, $y_2 = 0$, если допрашиваемый человек сказал правду, и $y_1 = 0$, $y_2 = 1$, если он лжет. Можно также использовать персептрон с одним единственным выходом, значение которого $y = 1$, если человек говорит правду, и $y = 0$, если он лжет.

Выборка обучающих примеров формируется в результате предварительных бесед следователя с подследственным, в ходе которых следователь задает вопросы, ответы на которые ему известны. Таким образом, следователь во время этих бесед снимает с подследственного векторы обучающих примеров $\boldsymbol{X}_q$ и $\boldsymbol{D}_q$.

Накопив достаточное число примеров, можно научить персептрон делать заключения о правильности показаний подследственного. Причем, поскольку персептрон обучился на примерах, сформированных самим подследственным, то можно полагать, что заключение персептрона будет объективно учитывать индивидуальные особенности организма допрашиваемого.

Приведенный способ создания настраиваемого детектора лжи обладает недостатком, состоящим в трудоемкости формирования обучающей выборки. Поэтому его можно рекомендовать для ответственных случаев, когда затраты на длительные предварительные беседы следователя с подследственным оправданы. В других случаях можно рекомендовать персептрон, настроенный на некоторого усредненного человека. Такой детектор лжи получается при использовании в качестве обучающей выборки ответов нескольких человек, относящихся к различным психологическим типам.

3.2.5. Нейросеть-антихакер

В средствах массовой информации появились сообщения о ряде успешных полицейских операций по выявлению и задержанию хакеров, пытавшихся взламывать запрещенные для широкого круга пользователей компьютерные системы. В качестве инструмента, позволившего обнаружить аномальную сетевую активность, вызываемую действиями хакеров, указывались нейросети. Рассмотрим, в чем состоит принцип действия таких нейросетей.

Прежде всего отметим, что поведение хакера, пытающегося взломать компьютерную программу, несколько отличается от поведения обычного законопослушного пользователя. Иногда хакер чаще, чем обычный пользователь, ударяет по одной и той же кнопке клавиатуры. Существуют приемы взламывания программ, связанные с определенными траекториями движения курсора мыши, например с многократным перемещением курсора по одной и той же области экрана. Поэтому, измерив параметры, характеризующие стиль работы различных пользователей, можно сформировать обучающую выборку соответствующих примеров и обучить нейросеть реагировать на различные сетевые отклонения, т.е. выявлять аномалии сетевой активности. Параметрами, характеризующими стиль работы пользователей, их портрет, могут быть: число загружаемых одновременно программ, скорость ударов по клавиатуре и мыши в единицу времени, частота повторения ударов

по одним и тем же клавишам, характер пользования мышью и др. Число этих параметров определяет размер входного вектора X и, соответственно, число нейронов входного слоя персептрона. На выходе персептрона целесообразно оставить один нейрон, значение которого $y = 0$ будет означать, что за компьютером находится обычный пользователь, а $y = 1$ — что пользователь хакер. Затем необходимо создать достаточный для обучения персептрона набор портретов хакеров и обычных пользователей. Далее нужны обычные действия по проектированию и обучению персептрона. Естественно, что качество нейросети-антихакера будет зависеть от того, насколько высокой окажется квалификация хакеров, приглашенных для создания обучающих примеров.

3.2.6. Нейросети в банковском деле

Банкротство фирм, кредитуемых банками, невозможность возврата ими кредитных средств не раз являлись причиной кризисов и банкротств весьма солидных банков. Поэтому вопрос о том, какова степень кредитного риска, каким клиентам опасно выдавать кредиты, а каким нет, для любого банка является одним из самых главных вопросов стабильности его существования.

Обычно клиенты банка — это частные лица и фирмы, занимающиеся различного рода бизнесом. Банки выдают кредиты под проценты, которые являются немаловажной статьей их доходов. Прежде чем принять решение о выдаче кредита, банкиры тщательно изучают и анализируют бизнес-план кредитуемого проекта. В бизнес-плане обычно указывается, куда будут расходоваться кредитные деньги (приобретение сырья, оборудования, производственного помещения, аренда и покупка транспортных средств и др.). Далее в бизнес-плане рассчитываются и указываются сроки и объемы ожидаемой прибыли, сроки выплат процентов и возврата всего объема кредитных денег банку. Помимо этого, банкирам предоставляются всевозможные сведения о фирме-клиенте. Указываются дата создания и место регистрации фирмы, ее численный состав, количество филиалов, средний возраст работников, уровень их образования, вид деятельности, обороты и их динамика, имущество, недвижимость, транспортные средства и пр.

Эксперты кредитного отдела банка, изучив всю информацию о фирмах, желающих получить кредит, дают свое заключение по каждому проекту. Окончательное же решение о возможности выдачи кредита обычно принимает руководитель банка, учитывая мнения экспертов, руководствуясь своим опытом и интуицией.

Однако в последнее время руководители многих английских банков стали спрашивать мнение еще и у нейронной сети. Рассмотрим общие принципы создания и работы такой сети. Совет-

чиком английских банкиров является обычный многослойный персептрон. На его вход подается вектор X, в котором кодируются данные о фирме, подавшей заявку на получение кредита. Такими данными являются: время жизни фирмы, вид ее деятельности, средний возраст, пол, численность работников, уровень их профессионализма, число филиалов фирмы, всевозможные экономические показатели за несколько лет. Персептрон имеет один выходной нейрон, который по многобалльной шкале вычисляет степень живучести, а следовательно, и платежеспособности фирмы.

Персептрон — советчик банкиров обучался на примерах, взятых из собранных с английской тщательностью архивных материалов нескольких банков, в которых был отражен многолетний опыт их кредитной деятельности: параметры фирм, некогда получавших кредиты, и, самое главное, результат сотрудничества с банком — своевременность выплаты процентов и возврата кредита. По свидетельству банкиров, применивших нейронную сеть, она помогла им выявить ряд потенциальных неплательщиков и скорректировать финансовую политику банков.

3.2.7. Прогнозирование валютных курсов и котировок ценных бумаг

Прогнозирование — это одна из самых востребованных задач, возникающих в различных областях человеческой деятельности. Задача прогнозирования в общем случае состоит в получении будущих значений каких-либо параметров на основе анализа имеющихся значений этих параметров. Обычно речь идет о прогнозировании *временнóго ряда*, т.е. совокупности значений прогнозируемого параметра на некотором интервале времени $[T(n + 1); T(n + k)]$, где k — интервал прогнозирования. При этом $T(n)$ — текущий момент времени. Часто возникает необходимость предсказать не значения самого временнóго ряда, а вероятность того или иного характера его поведения на заданном интервале, т.е. будет ли он возрастающим, убывающим или прогнозируемый параметр будет находиться в определенных пределах.

Существуют различные подходы к решению задач прогнозирования — от построения сложнейших нестационарных математических моделей, учитывающих физические, химические, биологические и другие законы природы, до статистических методик поиска зависимостей прогнозируемых параметров от времени. Однако в последнее время с традиционными подходами успешно конкурируют нейросетевые технологии, которые особенно хорошо себя зарекомендовали при прогнозировании финансовых рынков: котировок ценных бумаг и валютных курсов, а также общих экономических индексов, таких как индекс Доу Джонса и др.

Рассмотрим пример применения нейросети для прогнозирования на три дня вперед курса американского доллара по отношению к российскому рублю. На рис. 3.15 представлена номограмма, изображающая изменения курса доллара США с марта по май 2003 г. Выборку обучающих примеров можно сформировать с использованием метода скользящих окон, согласно которому выбирается временнóй интервал, например 45 дней — с 4 марта до 17 апреля. В качестве x_1 задается курс доллара, который был 4 марта, x_2 — курс на 5 марта и так далее, в качестве x_{45} — курс на 17 апреля. В качестве желаемого выхода сети d_1 принимается курс доллара на 18 апреля, d_2 — курс на 19 апреля, d_3 — курс на 20 апреля. Таким образом будет сформирован первый обучающий пример $\boldsymbol{X}_1 - \boldsymbol{D}_1$ для персептрона, имеющего 45 нейронов входного слоя и 3 нейрона выходного слоя.

Для формирования второго обучающего примера сдвинем окно на одну позицию (один день) вправо и выполним аналогичные операции. Перемещая окно Q раз, мы получим выборку из Q обучающих примеров. Далее следует определиться с количеством внутренних слоев и нейронов персептрона (см. подразд. 3.3) и выполнить обучение, например методом обратного распространения ошибки. Проверить качество получившейся прогностической программы, оценить точность прогноза можно на тестовых выборках, которые легко получить, располагая окна таким образом, чтобы в них не попадали использованные при обучении даты. Программа выполнит прогноз на три дня вперед, если окно расположить в самом конце номограммы, т. е. таким образом, чтобы в качестве x_{45} был курс доллара на сегодняшний день.

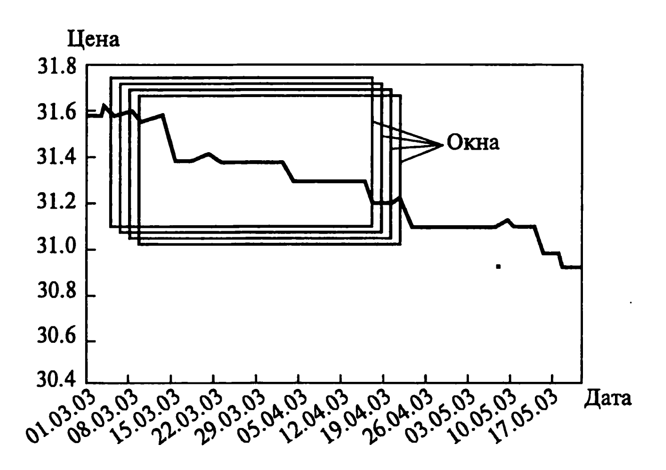


Рис. 3.15. Номограмма изменения курса американского доллара к российскому рублю за март—май 2003 г.

Как известно, качество прогностических программ зависит от полноты учета факторов, влияющих на прогнозируемый параметр. На курс доллара США влияет политическая и экономическая обстановка в Америке, России, странах Европы, Азии и других континентов, наличие или отсутствие военных конфликтов на планете, президентских выборов, террористических актов, стихийных бедствий. На курс доллара влияют также погодно-климатические изменения, эффект начала и конца рабочей недели, месяца, года, наличие праздников. Наконец, на курс доллара оказывают влияние явления космической природы. Всплески солнечной активности, магнитные бури вызывают изменение самочувствия, психологического состояния, жизненной активности многих людей. Естественно, это сказывается на экономическом состоянии отдельных фирм, регионов, стран, подвергшихся космическим воздействиям. Барометром, фиксирующим такие изменения, служат курсы валют и котировки акций, общие экономические индексы. Все они являются функциями огромного количества переменных, и их значения формируются в результате совместного действия множества разнообразных зависимостей и закономерностей, выявить и изучить которые традиционными способами не представляется возможным.

Рассмотрим, как эти вопросы можно решать с помощью нейросетевых технологий. Например, чтобы учесть эффект начала и конца рабочей недели, можно добавить в персептроне один входной нейрон, в качестве входного сигнала которого задать единицу, если прогнозируемый день является понедельником, двойку, если он приходится на пятницу, и ноль при любом другом дне недели. Аналогично, путем введения новых входных нейронов, можно учесть баллы солнечной активности, фазы Луны, эклиптические долготы планет, их положения в знаке, терме, фасе, их скорости и склонения и другие астрономические и астрологические события, приходящиеся на рассматриваемый период времени. Естественно, что многие из этих параметров не являются *значащими*, т.е. не влияют на интересующие нас прогнозируемые величины. Решить вопрос о значимости того или иного фактора можно, опять же, с помощью нейросети, наблюдая за тем, приводит ли введение соответствующего входного нейрона к уменьшению ошибки прогноза на тестовых примерах.

Подводя итог изложенному, отметим, что в настоящее время брокерами, финансовыми игроками, экономистами, бизнесменами разных стран активно используются нейросети, учитывающие множество факторов, в том числе и астрологических.

Нейросети стали незаменимым инструментом для выявления и использования скрытых, не известных ранее и никогда не исследованных закономерностей, в результате чего многие науки, называвшиеся ранее неофициальными, например финансовая

астрология, получили мощный толчок для теоретических и прикладных исследований.

3.2.8. Задачи, решаемые с помощью нейросетей

Список примеров практического применения нейросетевых технологий можно во много раз увеличить. Однако и те немногие приведенные выше примеры убедительно показывают, что мы имеем универсальный и весьма эффективный инструмент для построения математических моделей самых разнообразных физических, технических, химических, экономических, социальных и другого рода объектов, процессов, явлений. Исследуя эти модели, мы можем решать широкий круг разнообразных практических задач. Так, если нам удалось построить математическую модель какого-то сложного технологического процесса, например, выплавки стали в мартеновской печи, или крекинга нефти в химическом реакторе, или производства электроэнергии на атомной электростанции, то, исследуя математическую модель, изучая влияние входных параметров на выходные, можно решить задачу *оптимизации* моделируемого технологического процесса. Это значит, что можно подобрать оптимальное сочетание входных параметров, обеспечивающих максимально высокое качество выплавляемой стали, рассчитать наиболее благоприятный ход химической реакции крекинга нефти, выбрать наиболее эффективный режим работы атомной станции.

Аналогично решаются задачи оптимизации в сфере бизнеса, экономики. В этом случае выходом нейронной сети может быть некая целевая функция, имеющая смысл экономической эффективности предприятия, валового продукта, прибыли или рентабельности фирмы.

Если математическая модель является нестационарной, т. е. составлена с учетом фактора времени, то ее можно использовать для решения задач *прогнозирования*. Это значит, что с помощью математической модели можно узнать, какими будут технологические, экономические, социальные, политические и другие показатели моделируемого объекта в будущем и как на них можно повлиять, принимая те или иные меры сегодня.

Если математическая модель работает в реальном режиме времени, т. е. оперативно получает сведения о текущих изменениях параметров моделируемого объекта, если результаты математического моделирования могут быть оперативно переданы оператору, управляющему объектом, или могут быть непосредственно введены в приборы, например, дозирующие подачу кокса, руды, кислорода и других химических компонентов в мартеновскую печь либо управляющие параметрами работы ядерного реактора, то

такая математическая модель будет решать задачу *управления* моделируемым объектом или процессом.

Помимо перечисленных задач оптимизации, прогнозирования и управления персептрон, как было показано выше, может решать задачи *распознавания и классификации образов*, причем под образами понимаются зрительные изображения, символы, тексты, запахи, звуки, шумы.

Отметим, что во всех примерах построения математических моделей с помощью нейросетевых технологий не требовалось знание и использование законов природы. Вместо этого нужно было подготовить обучающую выборку, содержащую статистические данные о предметной области. Если эта выборка оказывается достаточно *репрезентативной* (представительной), то нейросеть сама извлекает закономерности, необходимые для формирования математической модели, адекватной рассматриваемой предметной области. В этом отношении методика построения нейросетевых моделей напоминает методику построения регрессионных моделей. Последние, как известно, основаны на методе наименьших квадратов, позволяющем получать математические формулы, аппроксимирующие статистические данные. Однако в отличие от регрессионных, нейросетевые технологии представляют собой значительно более мощный и универсальный математический аппарат. Кроме того, не надо забывать, что в его основе лежит не просто математический трюк, а глубокий физический, психологический и общефилософский смысл, о котором достаточно много говорилось в начале данной главы.

3.2.9. Невербальность и «шестое чувство» нейросетей

Нейронная сеть (нейросеть) — это система, выполненная по образу и подобию человеческого мозга. Естественно, что она наследует его свойства, причем как положительные, так и отрицательные.

Как и человеческий мозг, нейросеть способна извлекать знания из данных, обнаруживать скрытые в них закономерности. Но, как и человек, нейросеть не способна объяснять, как она это делает.

Архимед открыл свой закон, лежа в ванне, Ньютон — наблюдая за падением яблока. Тот и другой гениальным образом догадались, нашли истину, не объяснив, как они это сделали.

Нейросеть, извлекая знания из данных, тоже способна выводить закономерности, делать догадки, открывать законы природы. Но, так же, как и человек, нейросеть не способна к четкой формулировке пунктов алгоритма, позволившего сделать то или иное умозаключение.

Согласно современным эзотерическим теориям, гений в момент озарения считывает информацию из параллельных миров.

Нелепо было бы утверждать, что нейросеть в момент извлечения знаний из данных проделывает аналогичную операцию. Поэтому моделирование деятельности мозга дает нам основания возразить против эзотерической теории познания. По-видимому, многое из того, что демонстрируют экстрасенсы, контактеры, шаманы, можно объяснить свойством мозга извлекать информацию из неполных данных — свойством, которое удается моделировать с помощью нейронных сетей.

Известны случаи, когда нейросети демонстрируют феномен, называемый в жизни шестым чувством. Они с успехом извлекают знания из анализа информации, из которой, казалось бы, эти знания извлечь невозможно. В подразд. 3.2.3 был приведен один из таких примеров, касающийся диагностики неисправностей авиационных двигателей по совокупности их полетных параметров. Феномен заключается в том, что помимо всех прочих нейросеть диагностирует и такие неисправности, которые традиционными диагностическими методами, основанными на законах физики, выявить не представляется возможным. Так, дефект «стружка в масле» считается побочным и обнаруживается только после вскрытия авиадвигателя на испытательном стенде. Этот дефект не выявляется традиционными диагностическими методами, поскольку его наличие, по мнению специалистов, никак не влияет ни на один снимаемый с авиадвигателя полетный параметр. Тем не менее, несмотря на отсутствие какой-либо логической связи между этим дефектом и параметрами работы авиадвигателя, нейросеть обнаруживает скрытую от обычных (вербальных) методов диагностики закономерность и ставит правильный диагноз относительно наличия или отсутствия стружки в масле.

Феномены подобного рода в практике применения нейросетевых технологий не являются редкостью. Особенно часто они наблюдаются при исследовании достаточно сложных объектов, когда нейросети выявляют связи и закономерности, о существовании которых специалисты, создавшие объект, не знают. Выявляются даже такие взаимосвязи и взаимные влияния, которые при поверхностном рассмотрении противоречат здравому смыслу специалистов и становятся понятными и объяснимыми только после более тщательного изучения объекта.

Таким образом, мы вправе заявить о наличии у нейросетей свойства, обычно называемого в жизни шестым чувством, — способности принимать правильные решения, алгоритм принятия которых с точки зрения известных истин объяснить не представляется возможным.

Невербальность знаний и «шестое чувство» нейросетей — это качества, вытекающие из самой их природы. Нет ничего удивительного в том, что нейросети, представляющие собой модель человеческого мозга, наследуют его свойства. Человеческий мозг

по своей сути является невербальным объектом. В процессе эволюции тысячелетиями от мозга требовалось обрабатывать поступающую информацию, делать из нее выводы и принимать решения. Но при этом не требовалось давать каких-либо объяснений. Потребность в вербализации человеческих умозаключений появилась только в последнее тысячелетие, и далеко не каждый современный человек обладает способностью объяснять все свои поступки. Многие наши действия совершаются под влиянием эмоций, «шестого чувства», не имеющего логических объяснений. И, возможно, дальнейшая эволюция человеческого мозга приведет его к еще более совершенной структуре, в которой алгоритм принятия решений и сами знания будут храниться в прозрачной для понимания форме. Но пока мы моделируем мозг, в котором знания закодированы в виде матрицы сил синаптических связей. В отличие от экспертных систем, где имеется возможность проследить всю цепочку логического вывода, мы не можем спросить нейросеть, почему она пришла к тому или иному выводу. Естественно, что это является недостатком нейросетевых технологий, преодолением которого занимаются некоторые исследовательские группы. Так, в работах [6, 7] предложена методика вербализации нейросетевых знаний, заключающаяся в последовательном упрощении (редуцировании) сети до такой степени, при которой она становится прозрачной для понимания выполняемых ею действий. Авторы методики привели интересный пример вербализации знаний нейросети, обученной прогнозированию результатов выборов президента США. Сеть редуцировалась до тех пор, пока число входных сигналов каждого нейрона не уменьшилось до трех. В результате закономерность, которую нейросеть извлекла из обучающей выборки, удалось описать в виде логической формулы.

1. Президент США потерпит поражение на выборах, если его правление было «плохое» или «ситуация политически нестабильна».

2. Правление президента было «плохое», если верны хотя бы два из следующих высказываний: «Имеет место серьезная конкуренция при выдвижении от правящей партии», «Год выборов является временем спада или депрессии», «Правящий президент не произвел существенных изменений в политике».

3. «Ситуация политически нестабильна», если верны хотя бы два из следующих высказываний: «В год выборов активна третья партия», «Имеет место серьезная конкуренция при выдвижении от правящей партии», «Во время правления были существенные социальные волнения».

Следует однако заметить, что в других случаях редуцирование нейросетей не приводит к желаемому результату из-за существенного понижения качества их работы.

По мнению многих специалистов в области искусственного интеллекта, человеческий мозг, являясь самой сложной из изве-

стных в природе систем, «не хочет раскрывать своих тайн». Подводя итог приведенным рассуждениям, отметим, что последняя фраза может быть отнесена и к модели человеческого мозга — нейронным сетям. Но это совсем не значит, что их удивительными свойствами не следует пользоваться.

3.3. ПРОЕКТИРОВАНИЕ И ОБУЧЕНИЕ ПЕРСЕПТРОНОВ

3.3.1. Теоремы существования

Из предыдущего изложения следуют два важных вывода.

1. Подавляющее большинство всех прикладных задач, решаемых методом математического моделирования, сводится к нахождению некоторой сложной функции, осуществляющей многомерное преобразование вектора входных параметров X на вектор выходных параметров Y.

2. Универсальным инструментом построения такой функции являются нейросетевые технологии.

Естественно, возникают вопросы: всегда ли можно построить нейросеть, выполняющую преобразование, заданное любой обучающей выборкой, и каким требованиям эта нейросеть должна удовлетворять?

Чтобы ответить на эти вопросы, надо вспомнить, что каждый нейрон нейронной сети выполняет суперпозицию (суммирование) сигналов, поступающих от других нейронов, которые в тех других нейронах прошли через нелинейное (например, сигмоидное) преобразование. Вопрос о том, можно ли любую функцию многих переменных представить в виде суперпозиции функций меньшего количества переменных, интересовал математиков на протяжении нескольких последних веков. Так, в 1900 г. на Всемирном математическом конгрессе в Париже знаменитым немецким математиком Давидом Гильбертом были сформулированы 23 проблемы, которые он предложил решать математикам начинающегося XX в. Одна из этих проблем, под номером тринадцать, декларировала невозможность такого представления.

Однако последующие исследования показали, что 13-я проблема Д. Гильберта имеет иное решение. В результате многолетней дискуссии между советским академиком А. Н. Колмогоровым [19, 20] и его учеником В. И. Арнольдом [1] были получены фундаментальные теоретические результаты, свидетельствующие о принципиальной возможности представления непрерывных функций нескольких переменных в виде суперпозиции функций меньшего числа переменных. Затем Хехт-Нильсеном [56, 57] эти результаты были переработаны применительно к нейронным сетям. В частно-

сти, было доказано, что для любого множества пар отличных между собой входных и выходных векторов произвольной размерности $(\boldsymbol{X}_q, \boldsymbol{D}_q)$, $q = 1, ..., Q$ существует двухслойный персептрон с сигмоидными передаточными функциями и с конечным числом нейронов, который для каждого входного вектора $\boldsymbol{X}_q$ формирует соответствующий ему выходной вектор $\boldsymbol{D}_q$. Таким образом, была доказана принципиальная возможность построения нейросети, выполняющей преобразование, заданное любой обучающей выборкой различающихся между собой примеров, и установлено, что такой универсальной нейросетью является двухслойный персептрон (т.е. персептрон с одним скрытым слоем) с конечным числом нейронов и сигмоидными передаточными функциями.

Для определения необходимого числа нейронов в скрытых слоях персептрона была предложена формула, являющаяся следствием теорем Арнольда — Колмогорова — Хехт-Нильсена:

$$\frac{N_y Q}{1+\log_2 Q} \le N_w \le N_y \left(\frac{Q}{N_x}+1\right)\left(N_x+N_y+1\right)+N_y, \qquad (3.44)$$

где N_y — размерность выходного сигнала; Q — число элементов обучающей выборки; N_w — необходимое число синаптических весов; N_x — размерность входного сигнала.

Оценив с помощью этой формулы необходимое число синаптических весов, можно рассчитать число нейронов в скрытых слоях. Например, число нейронов скрытого слоя двухслойного персептрона

$$N = \frac{N_w}{N_x+N_y}. \qquad (3.45)$$

3.3.2. Проблемы и методы проектирования

Теоретически для построения нейросетевой модели любого сколь угодно сложного объекта достаточно использовать персептрон с одним скрытым слоем сигмоидных нейронов, число которых определяется формулами (3.44), (3.45). Однако в практических реализациях персептронов как количество слоев, так и число нейронов в каждом из них часто отличаются от теоретических. Иногда целесообразно использовать персептроны с бóльшим числом скрытых слоев. Такие персептроны могут иметь меньшие размерности матриц синаптических весов, чем двухслойные персептроны, реализующие то же самое преобразование.

Строгой теории выбора оптимального числа скрытых слоев персептронов пока нет. На практике же чаще всего используются персептроны, имеющие один или два скрытых слоя, причем число нейронов в скрытых слоях обычно колеблется от N_x до $3N_x$.

При проектировании персептронов необходимо понимать, что персептрон должен не только правильно реагировать на примеры, на которых он обучен, но и уметь *обобщать* приобретенные знания, т.е. правильно реагировать на примеры, которых в обучающей выборке не было. Чтобы оценить способность сети к обобщению, помимо обучающей выборки примеров $\boldsymbol{X}—\boldsymbol{D}$ в рассмотрение вводят некоторое количество тестовых примеров $\boldsymbol{X}_T—\boldsymbol{D}_T$, которые относятся к той же самой предметной области, но в процессе обучения не участвуют. После обучения вычисляют среднеквадратичную погрешность между прогнозом сети $\boldsymbol{Y}$ и желаемым выходом сети $\boldsymbol{D}$ или $\boldsymbol{D}_T$. Среднеквадратичная погрешность персептрона, вычисленная на обучающей выборке $\boldsymbol{X}—\boldsymbol{D}$, называется *погрешностью обучения*, обозначаемой ε, а вычисленная на тестовой выборке $\boldsymbol{X}_T—\boldsymbol{D}_T$ — *погрешностью обобщения*, обозначаемой ε_T. При увеличении числа нейронов внутренних слоев персептрона N погрешность обучения ε обычно падает, тогда как погрешность обобщения ε_T сначала падает, а затем, начиная с некоторого оптимального значения $N = N_0$, возрастает. Характерные кривые зависимости погрешностей обучения и обобщения от числа нейронов внутренних слоев персептрона приведены на рис. 3.16.

Поведение этих кривых легко объяснить, если воспользоваться аналогией с аппроксимацией набора данных полиномами методом наименьших квадратов. Как известно, задача аппроксимации состоит в том, чтобы подобрать полином, наиболее правильно отражающий характер закономерности, представленной графически точками предметной области. На рис. 3.17 точки, соответствующие некоторым параметрам предметной области, изображены в системе координат $x—y$ в виде двенадцати черных и белых кружков, причем точки, отмеченные черными кружками, исполь-

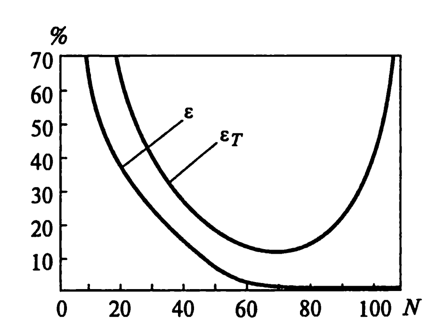


Рис. 3.16. Характерные зависимости погрешности обучения ε и погрешности обобщения ε_T от числа нейронов внутренних слоев персептрона

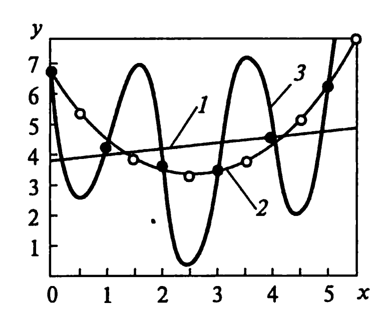


Рис. 3.17. Аппроксимация данных предметной бласти с помощью полиномов первого (кривая *1*), второго (кривая *2*) и пятого (кривая *3*) порядков

зовались при определении коэффициентов аппроксимирующих полиномов, а точки, отмеченные белыми кружками, не использовались, поэтому по ним можно судить о качестве аппроксимации. Выражаясь терминами нейросетевых технологий, черные кружки можно назвать обучающими примерами, а белые — тестирующими.

Кривая *1* изображает результат аппроксимации полиномом первой степени, кривая *2* — полиномом второй степени, *3* — полиномом пятой степени. Как видно из рис. 3.17, использование полинома первой степени не дает хорошей аппроксимации закономерности предметной области. Кривая, соответствующая полиному пятой степени, в точности проходит через все черные кружки, т.е. имеет нулевую погрешность обучения ε, тогда как тестовые белые кружки остаются в стороне.

Это означает, что свойством обобщения данная кривая не обладает. Наименьшую погрешность обобщения ε_T имеет кривая второго порядка, которая для данной предметной области является оптимальной.

Подводя итог проводимому сравнению, отметим, что нейросети, как и регрессионные модели, выстраивают функции, аппроксимирующие точки предметной области, причем для каждой решаемой задачи существует некоторое оптимальное значение степеней свободы математической модели. В случае полиномиальной аппроксимации степенями свободы являются определяемые коэффициенты (число которых на единицу больше порядка полинома). Число степеней свободы персептрона — это общее число синаптических весов (и порогов), которое определяется числом нейронов скрытых и выходных слоев. Свойство нейросети терять способность к обобщению при чрезмерном увеличении числа ее степеней свободы называют *переобучением*, или *гиперразмерностью нейросети.*

Итак, при проектировании персептронов существует проблема выбора необходимого числа нейронов. Число нейронов входного слоя персептрона должно совпадать с размерностью вектора входных параметров $\boldsymbol{X}$, который определен условиями решаемой задачи. Число нейронов выходного слоя должно совпадать с размерностью выходного вектора $\boldsymbol{Y}$, что также определено условиями задачи. Число скрытых слоев персептрона согласно теоремам Арнольда—Колмогорова—Хехт-Нильсена должно быть не менее одного. Число нейронов в скрытых слоях может быть приближенно оценено по формулам (3.40), (3.41), однако его желательно оптимизировать для каждой конкретной задачи.

Существуют два способа оптимизации числа нейронов в скрытых слоях — деструктивный и конструктивный. Деструктивный способ заключается в том, что первоначально строится сеть с заведомо избыточным числом степеней свободы, а затем в процессе

обучения из нее постепенно исключаются лишние синаптические связи и нейроны.

Существует довольно большое многообразие алгоритмов исключения избыточных степеней свободы нейросети. Наиболее простой из них состоит в обнулении тех синаптических весов, которые в процессе обучения нейросети стали значительно меньше их среднего значения. Однако малые значения синаптических весов не обязательно оказывают наименьшее воздействие на поведение нейронов и сети в целом. Поэтому более удачными признаются способы редукции сети, учитывающие ее чувствительность к структурным и параметрическим вариациям. Во многих современных нейропакетах запрограммированы эвристические алгоритмы, которые, производя последовательное исключение нейронов в скрытых слоях и отслеживая влияние таких исключений на погрешность обучения ε и погрешность обобщения ε_T, позволяют подобрать структуру сети, обладающую наилучшими эксплуатационными качествами.

Существуют также приемы, провоцирующие самостоятельное уменьшение значений весов с тем, чтобы исключить их, как только их величина опустится ниже установленного порога. Тенденцию к снижению синаптических весов можно вызвать, добавляя к целевой функции (3.12) слагаемые, штрафующие за большое значение веса. Такой штрафной функцией может быть слагаемое $\varepsilon' = \gamma\sum_{ij} w_{ij}^2$ или $\varepsilon'' = \frac{1}{2}\gamma\sum_{i,j}\frac{w_{ij}^2}{(1+\sum_k w_{ik}^2)}$. В случае использования второй штрафной функции тенденция к снижению наблюдается не у всех синаптических весов, а только у тех, которые действительно следует исключить [53].

Общим недостатком деструктивных алгоритмов является значительная длительность их работы, поскольку первоначальные вычисления производятся в сетях, имеющих избыточное количество нейронов. Этого недостатка лишены альтернативные конструктивные алгоритмы, которые предполагают постепенное добавление нейронов к сети, в которой их заведомо недостаточно. Новые нейроны добавляются каждый раз после определенного числа эпох обучения, а синаптическим весам и порогам новых нейронов присваиваются случайные числа. Поэтому после каждого добавления нового нейрона текущая погрешность обобщения нейросети ε_T резко увеличивается, но после нескольких эпох обучения становится меньше той, которая была до добавления нейрона. Однако, начиная с некоторого момента времени t_0, добавление новых нейронов перестает способствовать уменьшению ошибки ε_T, а, наоборот, приводит к ее увеличению, что свидетельствует о наступлении эффекта гиперразмерности (переобучения). Харак-

терная кривая зависимости погрешности обобщения от времени приведена на рис. 3.18. Очевидно, что в момент времени, предшествующий моменту t_0, структура нейросети была оптимальной для примеров данной предметной области.

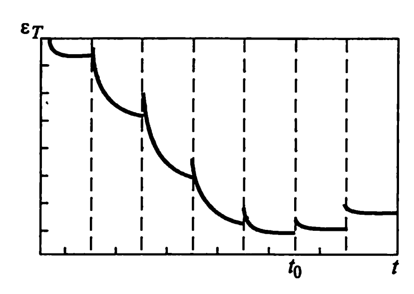


Рис. 3.18. Характерная кривая зависимости погрешности обобщения ε_T от времени t (штриховыми линиями отмечены моменты добавления новых нейронов)

Интересно отметить, что время обучения нейросети от начала до t_0 обычно оказывается лишь примерно в полтора раза больше, чем если бы в сети было сразу оптимальное число нейронов [11]. Эта цифра означает, что навыки, приобретенные нейросетью в процессе предварительного обучения, не теряются полностью при добавлении в нее нового нейрона.

Остается невыясненным вопрос, в каких конкретно местах нейросети следует добавлять новые нейроны. Их можно выбрать случайным образом или воспользоваться алгоритмами расщепления нейронов, приведенными в книге [22]. Согласно одному из таких алгоритмов следует вести наблюдение за тем, как изменяется вектор Δw_{ij} в процессе предъявления сети обучающих примеров. Если при наблюдении обнаружится такой нейрон, что одна группа обучающих примеров стремится изменить его вес (вектор w_{ij}) в одном направлении, а другая группа примеров — в другом, то такой нейрон логично расщепить на два. Первому из двух новых нейронов логично придать вес, отличающийся от веса исходного нейрона на величину коррекции, обусловленной первой группой примеров, а второму нейрону — вес, скорректированный с помощью второй группы примеров. Добавление новых нейронов в сеть таким способом вообще не приводит к скачкообразному увеличению функции ошибки сети, наблюдаемому на рис. 3.18. В этом случае число эпох, необходимых для нахождения оптимальной структуры сети и ее обучения, несколько снижается, однако усложнение алгоритма и дополнительные вычисления, связанные с обнаружением нужного нейрона и его расщеплением, часто сводят на нет указанный выигрыш.

3.3.3. Проблемы и методы обучения

Как было показано в предыдущих подразделах, изобретение алгоритма обратного распространения ошибки открыло путь широкому практическому применению многослойного персептрона.

Вместе с тем с расширением фронта научных исследований обнаружились и недостатки этого алгоритма.

Прежде всего отметим, что алгоритм обратного распространения ошибки в его первоначальном изложении реализовывал *метод наискорейшего спуска*, который является далеко не самым лучшим градиентным методом теории оптимизации. Эта теория ставит своей задачей поиск минимума некоторой целевой функции (функционала) ε, которая зависит от нескольких переменных, представленных в виде вектора $\boldsymbol{w} = [w_1, w_2, ..., w_n]^T$. В случае обучения многослойного персептрона целевая функция ε — это квадратичная ошибка персептрона, рассчитанная по формуле (3.12), а $\boldsymbol{w}$ — вектор синаптических весов межнейронных связей.

Все градиентные методы теории оптимизации основаны на разложении целевой функции $\varepsilon(\boldsymbol{w})$ в ряд Тейлора в окрестности некоторой начальной точки $\boldsymbol{w}$ n-мерного пространства переменных:

$$\varepsilon(\boldsymbol{w} + \boldsymbol{p}) = \varepsilon(\boldsymbol{w}) + [\boldsymbol{g}(\boldsymbol{w})]^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \boldsymbol{H}(\boldsymbol{w})\boldsymbol{p} + \dots, \qquad (3.46)$$

где $\boldsymbol{p}$ — вектор, вдоль которого строится разложение в ряд Тейлора, $\boldsymbol{g}(\boldsymbol{w})$ — вектор градиента целевой функции $\boldsymbol{g}(\boldsymbol{w}) = \left[\frac{\partial\varepsilon}{\partial w_1}, \frac{\partial\varepsilon}{\partial w_2}, \dots, \frac{\partial\varepsilon}{\partial w_n}\right]^T$. Матрица

$$\boldsymbol{H}(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial^2\varepsilon}{\partial w_1 \partial w_1} & \dots & \frac{\partial^2\varepsilon}{\partial w_1 \partial w_n} \\ \dots & \dots & \dots \\ \frac{\partial^2\varepsilon}{\partial w_n \partial w_1} & \dots & \frac{\partial^2\varepsilon}{\partial w_n \partial w_n} \end{bmatrix},$$

составленная из производных второго порядка, называется матрицей Гессе, или *гессенианом*.

Разложение (3.46) можно считать квадратичным приближением целевой функции $\varepsilon(\boldsymbol{w})$ в окрестности точки $\boldsymbol{w}$ с точностью погрешности отсеченной части $O(h^3)$, где $h = \|\boldsymbol{p}\|$, т.е.

$$\varepsilon(\boldsymbol{w} + \boldsymbol{p}) = \varepsilon(\boldsymbol{w}) + [\boldsymbol{g}(\boldsymbol{w})]^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \boldsymbol{H}(\boldsymbol{w})\boldsymbol{p} + O(h^3). \qquad (3.47)$$

В теории оптимизации это разложение используется при построении алгоритмов поиска минимальной точки целевой функции. Полагая вектор $\boldsymbol{p}(t)$ направлением поиска в момент времени t, перепишем последнее представление в виде

$$\varepsilon\left(\boldsymbol{w}(t)+\boldsymbol{p}(t)\right)=$$

$$=\varepsilon(\boldsymbol{w}(t))+\left[\boldsymbol{g}(\boldsymbol{w}(t))\right]^T\boldsymbol{p}(t)+\frac{1}{2}\boldsymbol{p}(t)^T\boldsymbol{H}(\boldsymbol{w}(t))\boldsymbol{p}(t)+O(h^3). \quad (3.48)$$

Далее будем считать, что левая часть этого выражения есть значение целевой функции в новый момент времени, т.е. $\varepsilon(\boldsymbol{w}(t)+\boldsymbol{p}(t))=\varepsilon(t+1)$, тогда как $\varepsilon(\boldsymbol{w}(t))=\varepsilon(t)$. Если в разложении (3.48) ограничиться первыми двумя слагаемыми, то для выполнения условия уменьшения значения целевой функции со временем

$$\varepsilon(t+1)<\varepsilon(t) \quad (3.49)$$

необходимо, чтобы

$$\left[\boldsymbol{g}(\boldsymbol{w}(t))\right]^T\boldsymbol{p}(t)<0. \quad (3.50)$$

Последнее будет выполнено, если вектор $\boldsymbol{p}(t)$, определяющий направление минимизации, выбрать равным антиградиенту целевой функции, т.е.

$$\boldsymbol{p}(t)=-\boldsymbol{g}(\boldsymbol{w}(t)). \quad (3.51)$$

Таким образом, при задании вектора $\boldsymbol{p}(t)$ согласно формуле (3.51) итерационный процесс $\boldsymbol{w}(t+1)=\boldsymbol{w}(t)+\boldsymbol{p}(t)$ будет приводить к уменьшению значения целевой функции. Если теперь ввести коэффициент η, влияющий на шаг итераций, то получим

$$\boldsymbol{w}(t+1)=\boldsymbol{w}(t)+\eta\boldsymbol{p}(t), \quad (3.52)$$

т.е. придем к методу наискорейшего спуска, реализованному в алгоритме обратного распространения ошибки (см. подразд. 3.1.7).

Согласно этому алгоритму процесс поиска минимума функции $\varepsilon(\boldsymbol{w})$ можно представить следующим образом. В некоторой случайно заданной точке поверхности ошибок находится направление скорейшего спуска (антиградиента), затем делается шаг вниз на расстояние, пропорциональное крутизне склона (градиенту) и коэффициенту скорости обучения η. В новой точке снова определяются направление и величина антиградиента, согласно которым делается следующее перемещение и т.д. Это значит, что при виде сверху на поверхность, изображающую целевую функцию, каждое такое перемещение производится в направлении, ортогональном к проходящей через данную точку линии постоянного уровня (изолинии).

Если бы изолинии поверхности ошибок нейросети представляли собой концентрические окружности, как показано на рис. 3.19, *а*, то направление антиградиента указывало бы на точное расположение точки минимума целевой функции. Однако поверхность ошибок имеет более сложный характер. Так, на рис. 3.19, *б* изолинии поверхности ошибок имеют вид эллипсов, а сама поверхность вблизи минимальной точки имеет форму оврага. В этом слу-

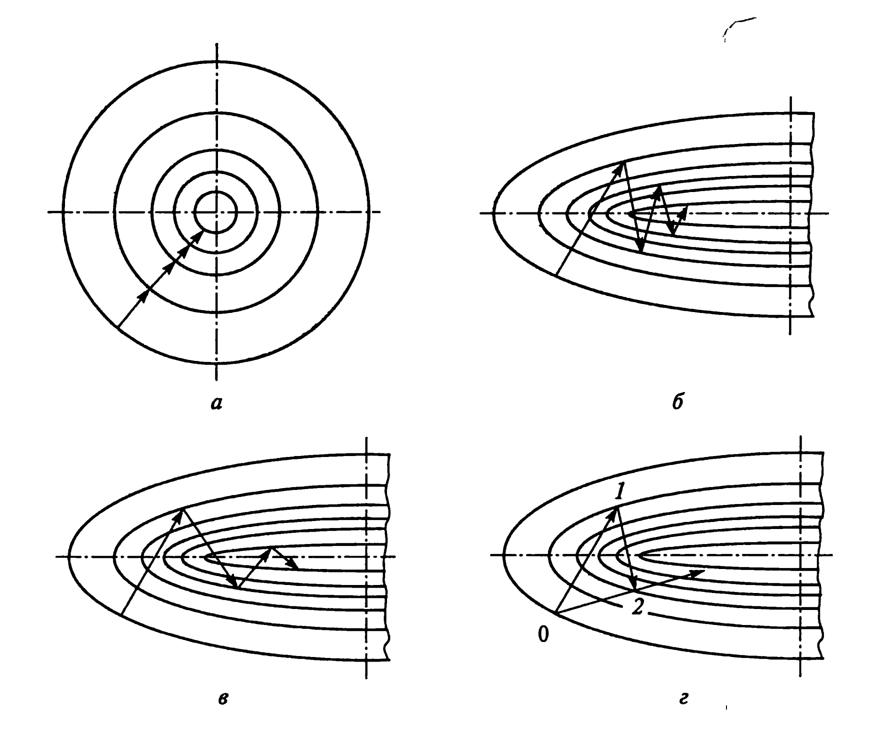


Рис. 3.19. Движение точки наблюдения по поверхности ошибок нейросети, обучаемой методом обратного распространения ошибки (*а*, *б*), методом обратного распространения ошибки с учетом инерции (*в*), ParTan-методом (*г*)

чае траектория градиентного спуска представляет собой ломаную линию, каждый отрезок которой ортогонален к линии уровня в той точке поверхности ошибок, из которой производится очередной шаг.

Наблюдая за этим процессом, можно представить себе, что по склону оврага спускается некое животное, которое каждый раз прыгает вниз в сторону максимальной крутизны поверхности оврага. Очевидно, что прыжки животного были бы более рациональными, если бы оно прыгало, считаясь с инерцией своей массы, т.е. каждый раз изменяло свою траекторию не так резко, как это предписывает направление антиградиента, а стремилось бы сохранить прежнее направление движения, как изображено на рис. 3.19, *в*. Указанное свойство инерции можно смоделировать, если в формуле для коррекции каждого весового коэффициента добавить слагаемое, пропорциональное величине коррекции этого коэффициента на предыдущем шаге:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) + \eta \boldsymbol{p}(t) + \alpha(\boldsymbol{w}(t) - \boldsymbol{w}(t-1)), \qquad (3.53)$$

где α — *коэффициент инерции* (*импульса* или *момента*), обычно задаваемый из интервала [0,1].

Другим способом учета инерции движения точки по поверхности целевой функции является *метод сопряженных градиентов*:

$$\boldsymbol{p}(t) = -\boldsymbol{g}(t) + \beta(t-1)\boldsymbol{p}(t-1). \qquad (3.54)$$

Здесь $\boldsymbol{g}(t) = \boldsymbol{g}(\boldsymbol{w}(t))$ — антиградиент; $\beta(t-1)$ — коэффициент сопряжения, обычно определяемый по формуле

$$\beta(t-1) = \frac{\boldsymbol{g}^T(t)(\boldsymbol{g}(t) - \boldsymbol{g}(t-1))}{\boldsymbol{g}^T(t-1)\,\boldsymbol{g}(t-1)}. \qquad (3.55)$$

Суть еще одного приема, называемого *ParTan-методом*, продемонстрирована на рис. 3.19, *г*. Его идея заключается в том, что выполняются два или несколько шагов в сторону антиградиента, причем координаты точек функции ошибок перед первым шагом и после последнего шага запоминаются. Затем делается шаг в направлении, соединяющем первую и последнюю запомненные точки.

Существует группа методов, называемых *квазиньютоновскими*, в которых помимо первых производных используются еще и вторые производные целевой функции. Квази-ньютоновский алгоритм получается при сохранении трех слагаемых разложения целевой функции в ряд Тейлора (3.48). Продифференцировав их по $\boldsymbol{p}(t)$ и приравняв к нулю, получим:

$$\boldsymbol{g}(\boldsymbol{w}(t)) + \boldsymbol{H}(\boldsymbol{w}(t))\boldsymbol{p}(t) = 0. \qquad (3.56)$$

Отсюда

$$\boldsymbol{p}(t) = -\left[\boldsymbol{H}(\boldsymbol{w}(t))\right]^{-1}\boldsymbol{g}(\boldsymbol{w}(t)). \qquad (3.57)$$

Таким образом, используя итерационную формулу (3.52), мы получаем алгоритм, подразумевающий вычисление градиента $\boldsymbol{g}$ и гессениана $\boldsymbol{H}$ на каждом итерационном шаге. Однако в практических реализациях вместо гессениана используются его аппроксимации.

Существует группа методов, не требующих вычисления производных от целевой функции и потому называемых *неградиентными*. Однако, как правило, они значительно уступают по эффективности градиентным методам и поэтому не рекомендуются для обучения нейросетей.

Существуют также методы обучения нейросетей, называемые *эвристическими*. Как правило, они не имеют строгого теоретического обоснования, но в них отражается личный опыт работы авторов в области нейросетевых технологий.

Во всех рассмотренных здесь методах обучения нейросетей присутствует проблема выбора параметра η, определяющего длину шага вдоль выбранного направления оптимизации $\boldsymbol{p}(t)$. Простей-

ший способ состоит в фиксации постоянных значений η на весь период обучения. При этом рекомендуется величину η задавать раздельно для каждого слоя персептрона, например по эмпирической формуле [60]:

$$\eta \le \min\left(\frac{1}{n_i}\right), \tag{3.58}$$

где n_i — число входов i-го нейрона в слое.

Другие методики предполагают динамическое изменение η в ходе обучения в зависимости от поведения целевой функции ошибок ε, вычисляемой по формуле (3.42). Для более быстрой сходимости коэффициент η стремятся увеличивать по мере снижения функции ошибок, однако не допуская ее существенного возрастания.

В настоящее время основная проблема обучения персептронов состоит в том, что поверхность функции ошибок обычно имеет очень сложную форму со множеством локальных минимумов. Поэтому все изложенные выше методы обычно приводят к одному из локальных минимумов, лежащих в окрестности начальной точки обучения. Если после нахождения такого минимума погрешность обучения нейросети признается неудовлетворительной, то сеть «встряхивают», давая весовым коэффициентам случайные приращения, и продолжают процесс обучения из другой точки. Часто процесс обучения приобретает характер длительного экспериментирования, в ходе которого пробуются различные оптимизационные алгоритмы с различными параметрами. В результате успех применения нейросетевых технологий ставится в зависимость от опыта и интуиции специалиста, числа различных оптимизационных алгоритмов, имеющихся в его распоряжении.

В связи с этим актуальным является развитие методов *глобальной оптимизации*, т.е. таких, которые позволяют найти глобальный минимум многоэкстремальной целевой функции. Среди множества возможных подходов наиболее успешным признается идея *генетических алгоритмов*. Эта идея, впервые предложенная Дж. Холландом в 1970-х гг. [58], состоит в имитации природных оптимизационных процессов, происходящих при эволюции живых организмов.

Как известно, основы теории эволюции были сформулированы Чарльзом Дарвином в 1859 г. в его знаменитой работе «Происхождение видов путем естественного отбора». Согласно эволюционной теории природа оптимизирует все живое благодаря двум биологическим механизмам — естественному отбору и генетическому наследованию. Суть естественного отбора заключается в том, что наиболее приспособленные особи лучше выживают и приносят больше потомства, чем менее приспособленные. Механизм генетического наследования состоит в следующем. Почти в каж-

дой клетке любого живого организма имеется набор хромосом, несущих информацию об этом организме. Основная часть хромосомы — нить ДНК (молекула дезоксирибонуклеиновой кислоты), которая состоит из четырех видов соединений — нуклеотидов, идущих в определенной последовательности. Нуклеотиды обозначаются буквами А, Т, С и G, их порядок следования кодирует все генетические свойства организма.

Ген — это отрезок цепи ДНК, отвечающий за определенное свойство особи, например за цвет глаз, тип волос, цвет кожи и т. д. Установлено, что вся совокупность генетических признаков человека кодируется посредством примерно 60 тыс. генов, суммарная длина которых составляет более 90 млн нуклеотидов.

При размножении живых организмов происходит слияние двух родительских половых клеток: хромосомы родителей сближаются вплотную, затем их нити ДНК разрываются в нескольких случайных местах и хромосомы обмениваются своими частями. Таким образом, молекулы ДНК потомков случайным образом приобретают гены как отца, так и матери.

При наследовании возможны мутации — изменения генов в половых клетках одного из родителей. Измененные гены передаются потомку и придают ему новые свойства, отличные от свойств родителей. Если эти новые свойства окажутся полезными, т. е. потомок будет более приспособлен к окружающей среде, то в процессе естественного отбора он выживет и создаст новое более совершенное потомство. Таким образом, механизмы естественного отбора, изменчивости и наследственности являются источником совершенствования биологических видов, методом оптимизации свойств живых организмов, созданным самой природой.

Генетические алгоритмы, предназначенные для оптимизации (обучения) весовых коэффициентов нейронной сети, работают следующим образом. Сначала создается некоторая начальная популяция особей, каждая из которых имеет свою собственную «хромосому» — вектор весовых коэффициентов нейронной сети $\boldsymbol{w} = [w_1, w_2, ..., w_n]^T$. Для каждой особи вычисляется целевая функция $\varepsilon(\boldsymbol{w})$, являющаяся мерой приспособленности особи к существованию. Первоначальная популяция равномерно распределяется в пространстве оптимизируемых параметров. Таким образом, точки, соответствующие каждой особи, более-менее равномерно распределяются по поверхности целевой функции, как показано на рис. 3.20, *а*.

Отбор особей для *скрещивания*, необходимого для создания нового поколения, может основываться на различных принципах. Одним из наиболее распространенных считается принцип элитарности, согласно которому к скрещиванию допускаются наиболее приспособленные особи, а наихудшие отбраковываются и заменяются вновь создаваемым потомством. Этот процесс назы-

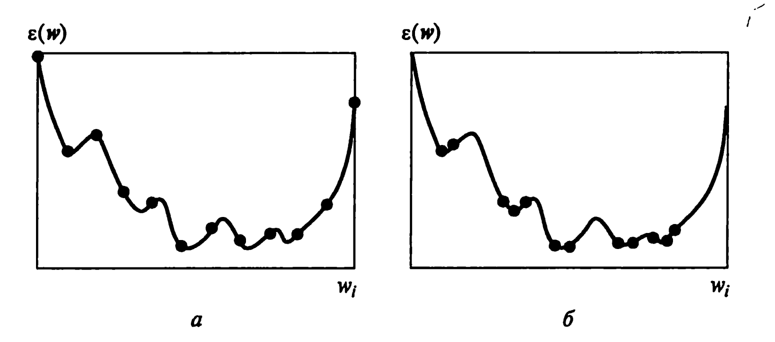


Рис. 3.20. Распределение по поверхности целевой функции точек, соответствующих хромосомам особей начальной популяции (*а*) и после *n* поколений (*б*)

вается *селекцией*. Обычно процесс скрещивания состоит в том, что хромосомы родителей случайным образом рассекаются на две неравные части, после чего они соединяются так, что хромосомы потомков содержат часть хромосомы отца и часть хромосомы матери, как показано на рис. 3.21. В ситуации, изображенной на рис. 3.21, после скрещивания хромосомы отца (фрагменты a_1 и a_2) с хромосомой матери (фрагменты b_1 и b_2) образовалась пара новых хромосом, первая из которых имеет фрагменты a_1 и b_2, а вторая — фрагменты b_1 и a_2.

Следующая генетическая операция называется *мутацией* и состоит в замене некоторого случайным образом выбранного элемента (гена) случайно выбранного вектора (особи) на новое, случайным образом заданное допустимое значение. Мутации обычно подвергается не более 1...5 % бит хромосом всей популяции. В результате всех этих генетических операций формируется новое поколение, число особей которого равно числу особей предыдущего поколения. Как показывает опыт, новые поколения, созданные в результате селекции, скрещивания и мутаций, в среднем имеют меньшие значения целевой функции (как показано на рис. 3.20, *б*), т.е. новые поколения являются более совершенными.

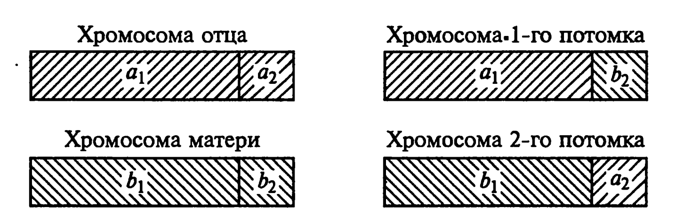


Рис. 3.21. Операция скрещивания, применяемая в генетических алгоритмах

Процесс смены поколений завершается после достижения заданного числа итераций или после того, как одна из особей приобретет заданное минимальное значение целевой функции. Эта особь является победителем, и ее хромосома принимается за окончательное решение генетического алгоритма.

При обучении персептронов обычно применяют различные вариации генетических алгоритмов, связанные с выбором параметров и способов селекции, скрещивания и мутаций. Эффективным оказалось совмещение генетических алгоритмов с ранее рассмотренными локальными алгоритмами оптимизации: на начальных стадиях работает генетический алгоритм, а затем особь-победитель или группа наиболее совершенных особей продолжают спуск к минимуму, например методом сопряженных градиентов.

В заключение отметим, что генетические алгоритмы, как и нейронные сети, восхищающие своей красотой и близостью к природным механизмам, являются новым перспективным разделом информатики. Они имеют свою независимую от нейросетевых технологий область применения и лежат в основе многих коммерческих пакетов, широко применяемых для решения разнообразных оптимизационных задач, возникающих в экономике, бизнесе, промышленности и других областях человеческой деятельности.

3.3.4. Подготовка входных и выходных параметров

Подбор обучающих примеров. От удачного подбора обучающих примеров во многом зависит успех создания нейронной сети, адекватно моделирующей предметную область. Прежде всего необходимо понимать, что не все параметры предметной области влияют на выходной вектор $\boldsymbol{Y}$. Например, знание температуры тела вряд ли поможет в постановке диагноза больному, если задача состоит в выявлении у него одних только психических отклонений.

Параметры, которые не оказывают влияния на вектор $\boldsymbol{Y}$, называют *незначимыми* для этого выходного вектора. Естественно, что незначимые параметры не следует включать в список параметров входного вектора $\boldsymbol{X}$.

Однако на практике часто бывает трудно и даже невозможно установить, какие из параметров предметной области являются значимыми, а какие нет. Поэтому на первом этапе в вектор $\boldsymbol{X}$ включают как можно больше параметров, избегая только те из них, незначимость которых представляется очевидной.

После первоначального создания и обучения нейронной сети, незначимые параметры могут быть выявлены двумя способами.

1. Путем анализа значений весовых коэффициентов входных нейронов. Если окажется, что у какого-либо входного нейрона

синаптические веса значительно меньше, чем у других нейронов, то этот входной нейрон скорее всего соответствует незначимому параметру вектора $\boldsymbol{X}$.

2. Путем возмущения значений входных параметров и анализа реакции сети на эти возмущения. Если сеть не реагирует или слабо реагирует на изменения значения какого-либо входного параметра, то этот параметр не является значимым.

После выявления и исключения входных нейронов, соответствующих незначимым параметрам, качество нейросети улучшается, так как снижается ее размерность. Однако надо понимать, что слишком малое число входных параметров может привести к тому, что нейросети не хватит данных для выявления требуемых от нее закономерностей предметной области.

Предобработка обучающих примеров и интерпретация ответов. Параметры, описывающие предметную область, могут иметь самый разнообразный характер. Это могут быть числа с различными диапазонами изменений, качественные характеристики, такие как цвет волос и глаз пациента, даты (число, месяц, год), графические объекты. Поскольку нейросеть в состоянии обрабатывать только числа, то вся нечисловая информация должна быть закодирована в числовом виде.

Числовую информацию, приготовленную для нейросетевой обработки, желательно масштабировать, т.е. выровнять диапазоны изменения величин, например, ограничив их интервалом [0, 1] или [–1, 1]. Сделать это можно с помощью простейшего линейного преобразования:

$$\tilde{x}_n = \frac{x_n - x_{n\min}}{x_{n\max} - x_{n\min}}(b - a) + a, \tag{3.59}$$

где x_n и $\tilde{x}_n$ — значения исходного и масштабированного n-го параметра предметной области, подаваемого на n-й входной нейрон нейросети; $[x_{n\min}, x_{n\max}]$ — реальный диапазон изменения n-го параметра; $[a, b]$ — приемлемый диапазон изменения входных сигналов.

Желаемые выходные сигналы персептрона должны быть также закодированы в приемлемой форме и масштабированы в приемлемом диапазоне $[a, b]$. Например, в подразд. 3.2.7 рассматривалось применение персептрона для прогнозирования курса доллара, который с марта по май 2003 г. изменялся в диапазоне [30.6, 31.8]. Это значит, что при формировании обучающего вектора $\boldsymbol{D}$ следует применить формулу масштабирования, аналогичную (3.59):

$$\tilde{d}_m = \frac{d_m - d_{m\min}}{d_{m\max} - d_{m\min}}(b - a) + a, \tag{3.60}$$

где d_m и $\tilde{d}_m$ — заданное и масштабированное значения m-й компоненты вектора $\boldsymbol{D}$; $d_{m\min} = 30.6$; $d_{m\max} = 31.8$. Обученный на такой выборке персептрон будет формировать выходной вектор $\tilde{\boldsymbol{Y}}$, содержащий значения курса доллара, приведенные к диапазону $[a, b]$. Поэтому к ним должно быть применено преобразование, обратное масштабированию (3.60):

$$y_m = \frac{\tilde{y}_m - a}{b - a}(d_{m\max} - d_{m\min}) + d_{m\min}. \tag{3.61}$$

Таким образом, персептрон можно применять для моделирования предметной области, описываемой числовыми параметрами любого диапазона. При формировании обучающей выборки входные и выходные параметры желательно масштабировать — преобразовать к приемлемому диапазону $[a, b]$. Естественно, что ответы персептрона после этого следует *интерпретировать* путем применения преобразования, обратного масштабированию.

Теперь рассмотрим возможности интерпретации ответов персептрона при решении задач классификации. К ним относятся рассмотренные выше задачи создания нейросетевого детектора лжи (см. подразд. 3.2.4), выявления хакеров (см. подразд. 3.2.5), постановки диагнозов сложных технических устройств (см. подразд. 3.2.3) и диагнозов заболеваний человека (см. подразд. 3.2.2). Во всех рассмотренных случаях персептроны строились таким образом, что каждому классу (каждому диагнозу) отводился свой выходной нейрон. Размерность обучающих векторов $\boldsymbol{D}$ совпадала с числом выходных нейронов, а его компонентам d_m задавалось значение 1, если для подготовленного вектора $\boldsymbol{X}$ имел место диагноз, за который «отвечает» m-й выходной нейрон, и 0, если диагноз другой. При появлении на входе персептрона нового вектора параметров, не встречавшихся в обучающей выборке, персептрон вычислял вектор $\boldsymbol{Y}$, который нужно интерпретировать с целью получения заключения о классификации нового входного объекта. Наиболее распространенный способ интерпретации состоит в том, что выходному сигналу m-го нейрона присваивается значение 1, если $y_m > (b - a)/2$, и 0 — в противном случае. Причем логично полагать, что чем больше значение y_m, тем более вероятна правильность постановки диагноза m-й болезни или неисправности, а чем меньше значение y_m, тем вероятнее отсутствие m-й болезни, неисправности и т. п. В этом случае говорят, что значение y_m можно расценивать как *функцию принадлежности* объекта какому-либо классу или как *меру уверенности* ответа персептрона.

Таким образом, мы видим, что персептрон дает нечеткий ответ, оценивая вероятность возможной ошибки. Подобно добросовестному врачу-диагносту, персептрон может ответить, что у больного наверняка есть инфаркт миокарда, однако с вероятностью 65 % он

подозревает, что у больного был порок сердца и на 90 % персептрон уверен, что ишемической болезни сердца у больного нет.

В заключение отметим, что помимо рассмотренных способов подготовки данных и интерпретации ответов нейросетей, существует множество других приемов [5—7, 21], преследующих аналогичные цели.

3.3.5. Виды активационных функций

В современных нейронных сетях наиболее часто применяются следующие виды активационных функций.

Пороговые активационные функции. В начале главы с помощью формул (3.1)—(3.3) была введена пороговая активационная функция нейрона, представленная на рис. 3.2. Если в уравнении (3.1) под S понимать разность между взвешенной суммой входных сигналов и пороговым значением нейрона, т.е. $S = \sum_{i=1}^{n} x_i w_i - \theta$, то активационная функция $y = f(S)$ будет иметь вид, показанный на рис. 3.22, *а*. Однако чаще в нейросетях применяется пороговая функция, симметричная относительно начала координат (рис. 3.22, *б*).

Линейные активационные функции. На рис. 3.23, *а* показан график линейной активационной функции $y = S$ с неограниченной областью изменения. Такие функции могут быть и с ограниченной областью изменения: $y = -1$ при $S < -1$; $y = S$ при $-1 \le S \le 1$; $y = 1$ при $S > 1$ (рис. 3.23, *б*).

Сигмоидные активационные функции. На рис. 3.24, *а* изображен график сигмоидной функции, заданной уравнением

$$y = \frac{1}{1 + e^{-S}}, \tag{3.62}$$

а на рис. 3.24, *б* — уравнением

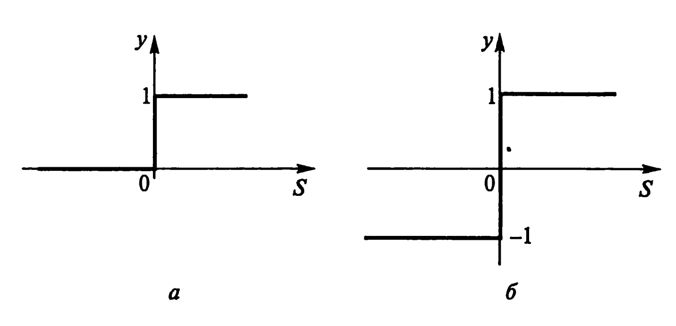


Рис. 3.22. Пороговые активационные функции с несимметричной (*а*) и симметричной (*б*) областями изменения

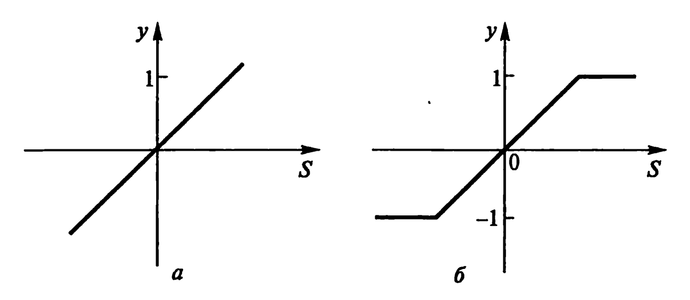


Рис. 3.23. Линейные активационные функции с неограниченной (*а*) и ограниченной (*б*) областями изменения

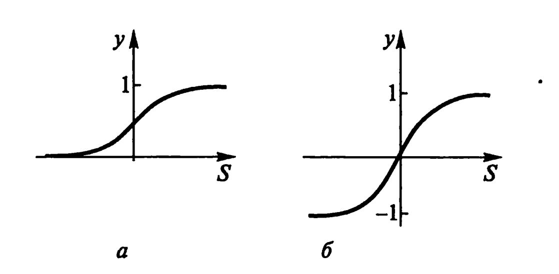


Рис. 3.24. Сигмоидные активационные функции с несимметричной (*а*) и симметричной (*б*) областями изменения

$$y = \frac{1 - e^{-S}}{1 + e^{-S}}. \qquad (3.63)$$

Аналогичный последнему графику характер имеют функции арктангенса $y = \frac{2}{\pi}\mathrm{arctg}S$ и гиперболического тангенса $y = \mathrm{th}S$, а также функция $y = \frac{S}{1+|S|}$, которые тоже называют сигмоидами.

Радиально-базисные активационные функции. В последнее время получают распространение нейросети, нейроны которых имеют активационные функции в форме функции Гаусса (рис. 3.25)

$$y = \mathbf{e}^{-\frac{S^2}{2\sigma^2}}, \qquad (3.64)$$

где S — евклидово расстояние между входным вектором $\boldsymbol{X}$ и центром активационной функции $\boldsymbol{C}$, $S = \|\boldsymbol{X} - \boldsymbol{C}\|$; σ — параметр гауссовой кривой, называемый *шириной окна.*

Такие активационные функции называют радиально-базисными (RBF), а соответствующие нейронные сети — RBF-сетями (см. подразд. 3.4).

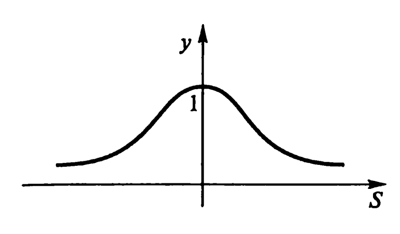


Рис. 3.25. Радиально-базисная активационная функция

Отметим, что все приведенные выше активационные функции, за исключением пороговой (см. рис. 3.22) и кусочно-линейной (рис. 3.23, *б*), являются непрерывно дифференцируемыми. Линейная функция (см. рис. 3.23, *а*) выполняет преобразование бесконечного входного множества значений переменных S в идентичное ему бесконечное множество переменных y. Пороговые активационные функции преобразуют множество S в бинарные множества $y = 0$ и $y = 1$ или $y = -1$ и $y = 1$. Остальные активационные функции преобразуют бесконечное входное множество S в ограниченные выходные множества: $y \in (0, 1)$, $y \in (-1, 1)$ и $y \in (0, 1]$. От вида используемых активационных функций зависят функциональные возможности нейросетей, а также способы их обучения.

3.4. РАДИАЛЬНО-БАЗИСНЫЕ СЕТИ

Радиально-базисной функцией (RBF) называется функция, радиально изменяющаяся вокруг некоторого центра, заданного вектором $\boldsymbol{C}$, и принимающая ненулевые значения только в окрестности этого центра. Ее аргументом является расстояние между текущим вектором $\boldsymbol{X}$ и вектором $\boldsymbol{C}$, т.е. $\varphi = \varphi(\|\boldsymbol{X} - \boldsymbol{C}\|)$.

В некотором смысле нейроны, имеющие радиально-базисные активационные функции являются логическим дополнением нейронов со ступенчатыми и сигмоидными активационными функциями. Действительно, нейрон Мак-Каллока—Питтса активизируется, когда $\sum_i x_i w_i - \theta \geq 0$, т.е. он имеет единичный (положительный) выход для точек пространства, лежащих по одну сторону гиперплоскости $\sum_i x_i w_i - \theta = 0$, и нулевой (отрицательный) — для точек, лежащих по другую сторону (рис. 3.26, *а*). Нейрон с радиально-базисной функцией также делит пространство входных параметров на две части, однако разделяющей поверхностью здесь является гиперсфера (рис. 3.26, *б*). Для точек пространства, лежащих внутри гиперсферы, выход нейрона положителен, а для точек, лежащих снаружи гиперсферы, он равен нулю (отрицателен).

Радиально-базисные нейроны обладают преимуществом, заключающимся в том, что с их помощью легче построить поверхность, обеспечивающую разделение входных параметров на клас-

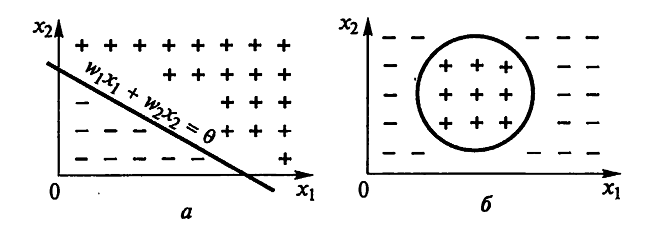


Рис. 3.26. Разделение пространства на две части нейроном Мак-Каллока-Питтса (*а*) и радиально-базисным нейроном (*б*)

сы. В связи с этим в радиально-базисных сетях отсутствует необходимость использования большого числа скрытых слоев. Типичная радиально-базисная сеть имеет только один скрытый слой, причем синаптические веса нейронов скрытого слоя равны единице, а нейроны входного и выходного слоев имеют линейные активационные функции. Как доказано в [53], такая сеть при достаточном числе нейронов скрытого слоя гарантирует решение любой задачи классификации образов.

Рассмотрим RBF-сеть, которая имеет N входов, один выход и J радиально-базисных нейронов скрытого слоя (рис. 3.27). Выборка обучающих примеров для такой сети состоит из Q входных N-мерных векторов $\boldsymbol{X}_q$, $q = 1, 2, ..., Q$ и соответствующих им выходных параметров d_q. Если число нейронов скрытого слоя J задать равным числу обучающих примеров Q, то работу RBF-сети, преобразующей входные векторы $\boldsymbol{X}_q$ в выходные параметры d_q, можно представить с помощью матричной операции:

$$\begin{bmatrix} \varphi(\|\boldsymbol{X}_1 - \boldsymbol{C}_1\|) & \varphi(\|\boldsymbol{X}_1 - \boldsymbol{C}_2\|) & \ldots & \varphi(\|\boldsymbol{X}_1 - \boldsymbol{C}_j\|) & \ldots & \varphi(\|\boldsymbol{X}_1 - \boldsymbol{C}_J\|) \\ \varphi(\|\boldsymbol{X}_2 - \boldsymbol{C}_1\|) & \varphi(\|\boldsymbol{X}_2 - \boldsymbol{C}_2\|) & \ldots & \varphi(\|\boldsymbol{X}_2 - \boldsymbol{C}_j\|) & \ldots & \varphi(\|\boldsymbol{X}_2 - \boldsymbol{C}_J\|) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \varphi(\|\boldsymbol{X}_q - \boldsymbol{C}_1\|) & \varphi(\|\boldsymbol{X}_q - \boldsymbol{C}_2\|) & \ldots & \varphi(\|\boldsymbol{X}_q - \boldsymbol{C}_j\|) & \ldots & \varphi(\|\boldsymbol{X}_q - \boldsymbol{C}_J\|) \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \varphi(\|\boldsymbol{X}_Q - \boldsymbol{C}_1\|) & \varphi(\|\boldsymbol{X}_Q - \boldsymbol{C}_2\|) & \ldots & \varphi(\|\boldsymbol{X}_Q - \boldsymbol{C}_j\|) & \ldots & \varphi(\|\boldsymbol{X}_Q - \boldsymbol{C}_J\|) \end{bmatrix} \times$$

$$\times \begin{bmatrix} w_1 \\ w_2 \\ \ldots \\ w_j \\ \ldots \\ w_J \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \ldots \\ d_q \\ \ldots \\ d_Q \end{bmatrix},$$

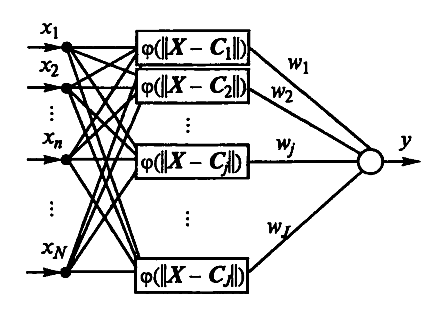


Рис. 3.27. RBF-сеть, имеющая N входов, один выход и J радиально-базисных нейронов скрытого слоя

которую перепишем в сокращенной матричной форме

$$\boldsymbol{\Phi}\, \boldsymbol{w} = \boldsymbol{d} \tag{3.65}$$

Искомый вектор весовых коэффициентов получается отсюда простым обращением матрицы радиально-базисных функций:

$$\boldsymbol{w} = \boldsymbol{\Phi}^{-1}\boldsymbol{d}. \tag{3.66}$$

Таким образом, для обучения RBF-сети не требуется итерационного процесса. Этот результат представляет теоретический интерес, однако практической ценности не имеет. Дело в том, что при большом числе обучающих примеров требование $J = Q$ приводит к вычислительным сложностям из-за чрезмерного увеличения числа нейронов внутреннего слоя. Кроме того, создаваемая нейросетью гиперповерхность, аппроксимирующая точки предметной области, в точности проходит через точки, изображающие обучающие примеры. Как показано ранее (см. подразд. 3.3.2), такая ситуация, называемая переобучением, или гиперразмерностью, ослабляет обобщающие свойства сети. Поэтому в практически используемых RBF-сетях число нейронов скрытого слоя выбирают значительно меньше числа обучающих примеров, т.е. $J << Q$. В этом случае матрица Ф не является квадратной, так как число строк Q в ней значительно больше числа столбцов J. Весовые коэффициенты w_j в этом случае могут быть определены из условия минимума квадратичной ошибки сети

$$\varepsilon = \sum_{q=1}^{Q}\left[\sum_{j=1}^{J} w_j \varphi\left(\left\|\boldsymbol{X}_q - \boldsymbol{C}_j\right\|\right) - d_q\right]^2. \tag{3.67}$$

В отличие от многослойного персептрона функция ошибки (3.67) не имеет локальных минимумов. Задача ее минимизации является линейной, и поэтому здесь применимы хорошо известные методы линейной оптимизации, которые сходятся на порядок быстрее, чем в случае обучения многослойного персептрона. Более того, задача определения весов $\boldsymbol{w}$ может быть решена путем псевдоинверсии прямоугольной матрицы Ф:

$$\boldsymbol{w} = \boldsymbol{\Phi}^{+}\boldsymbol{d}, \tag{3.68}$$

где $\boldsymbol{\Phi}^{+} = (\boldsymbol{\Phi}^{T}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^{T}$.

В качестве активационных функций в RBF-сетях чаще всего применяется функция Гаусса

$$\varphi(\|X - C_j\|) = \exp\left(-\frac{\|X - C_j\|^2}{2\sigma_j^2}\right). \quad (3.69)$$

Вид этой функции определяется двумя параметрами: вектором C_j, задающим ее центр, и скаляром s_j, задающим скорость ее убывания с ростом евклидова расстояния между центром C_j и текущей координатой X.

В случае, когда число нейронов скрытого слоя равно числу обучающих примеров ($J = Q$), центры активационных функций логично задать координатами векторов обучающей выборки, т.е. $C_j = X_q$ ($j = q = 1, ..., Q$), а ширины окон σ_j подобрать из тех соображений, чтобы часть пространства, в которой располагаются векторы X_q, была охвачена влиянием активационных функций. Например, в качестве σ_j можно задать евклидово расстояние от центра C_j до его ближайшего соседа либо среднеквадратичное расстояние до P ближайших соседей:

$$\sigma_j = \sqrt{\frac{1}{P}\sum_{p=1}^{P}\|C_j - C_p\|^2}, \quad (3.70)$$

где P рекомендуется задавать в интервале [3, 5].

После этого весовые коэффициенты w определяют по формуле (3.66).

В реальной же ситуации $J \ll Q$, поэтому предварительно выполняют *кластеризацию* — объединяют близкие между собой векторы X_q, называя такие объединения *кластерами*. Затем определяют центры кластеров, в которые и помещают центры активационных функций. Один из способов кластеризации векторов на заданное число кластеров рассмотрен в подразд. 3.6.

После фиксации центров активационных функций находят их ширины окон σ_j, например по формуле (3.70). Последний этап состоит в определении синаптических весов w путем минимизации функционала (3.67) или по формуле (3.68).

В заключение отметим, что в последнее время RBF-сети начинают успешно применять в задачах, которые традиционно решались нейронными сетями с сигмоидными функциями. Главным образом, это задачи распознавания и классификации, аппроксимации функций и прогнозирования. Интерес к RBF-сетям объясняется их следующими преимуществами:

RBF-сети имеют всего один скрытый слой, что избавляет конструктора сети от решения вопроса о числе слоев;

обучение RBF-сети сводится к решению линейной оптимизационной задачи, поэтому отсутствует опасность попадания в локальный минимум, а сам процесс обучения занимает на порядок меньше времени, чем процесс обучения многослойного персептрона.

Однако при проектировании RBF-сетей приходится решать вопрос об оптимальном числе нейронов скрытого слоя, выполнять кластеризацию входных обучающих векторов и определять ширины окон активационных функций.

3.5. РЕКУРРЕНТНЫЕ СЕТИ

Как показали нейрофизиологические исследования, мозг человека имеет гораздо более сложную структуру и механизмы взаимодействия между нейронами, чем те, которые реализованы в рассмотренных выше искусственных нейронных сетях. В частности, между биологическими нейронами выявлено большое число не только прямых, но и обратных связей. В связи с этим были предприняты попытки дополнить искусственные нейронные сети обратными связями, что привело к новым неожиданным результатам. Рассмотрим некоторые из них.

3.5.1. Рекуррентные сети на базе персептрона

На рис. 3.28, *а* приведен пример персептрона, у которого выходные сигналы y_1 и y_2 через элементы единичных задержек z^{-1} подаются обратно на входы персептрона. Таким образом, под воздействием входных сигналов x_1 и x_2 на выходе сети в момент времени t вырабатываются сигналы $y_1(t)$ и $y_2(t)$, а в следующий момент времени под воздействием этих сигналов, подаваемых на вход, вырабатываются новые выходные сигналы $y_1(t+1)$ и $y_2(t+1)$.

Нетрудно показать, что для всякой рекуррентной сети может быть построена идентичная сеть без обратных связей с прямым распространением сигнала (рис. 3.28, *б*), поэтому для обучения рекуррентных сетей может быть применен метод обратного распространения ошибки.

В настоящее время нашли применение рекуррентные нейросети, в которых элементы единичных задержек включены как в об-

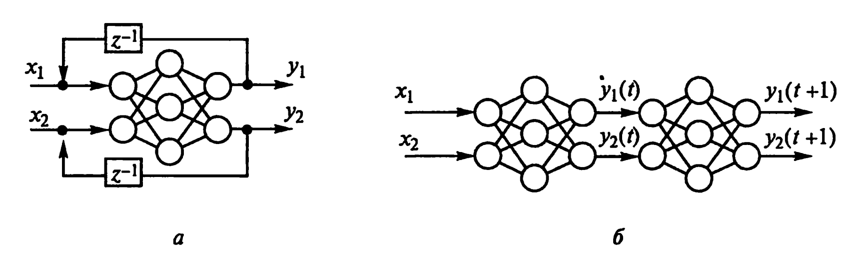


Рис. 3.28. Рекуррентная сеть на базе персептрона (*а*) и идентичный ей развернутый персептрон (*б*)

ратные, так и во входные связи, а сами обратные связи исходят как с выходных нейронов, так и с нейронов скрытых слоев. На рис. 3.29 приведена сеть, имеющая один вход и один выход, причем как входной, так и выходной сигналы подаются на нейроны скрытого слоя через элементы задержек. Таким образом, выходной сигнал, образующийся в момент времени $t + 1$, является функцией $N + P$ переменных:

$$y(t+1) =$$
$$= f(x(t),\ x(t-1), \ldots, x(t-(N-1)),\ y(t-1),\ y(t-2), \ldots, y(t-P)), \quad (3.71)$$

из которых N переменных представляют собой последовательность входных сигналов, а P переменных являются ответами персептрона в разные моменты времени и называются *контекстными аргументами*.

Такие нейронные сети удобно использовать для прогнозирования временны́х рядов. Например, если речь идет о задаче прогнозирования курса американского доллара, рассмотренной в подразд. 3.2.7, то в скользящем окне в качестве дат можно рассматривать N входных аргументов, а в качестве соответствующих значений курса доллара — P контекстных аргументов формулы (3.71). Шириной скользящего окна будет величина N, которую следует принять равной P. Значение функции $y(t + 1)$ тогда будет означать прогноз курса доллара на день вперед.

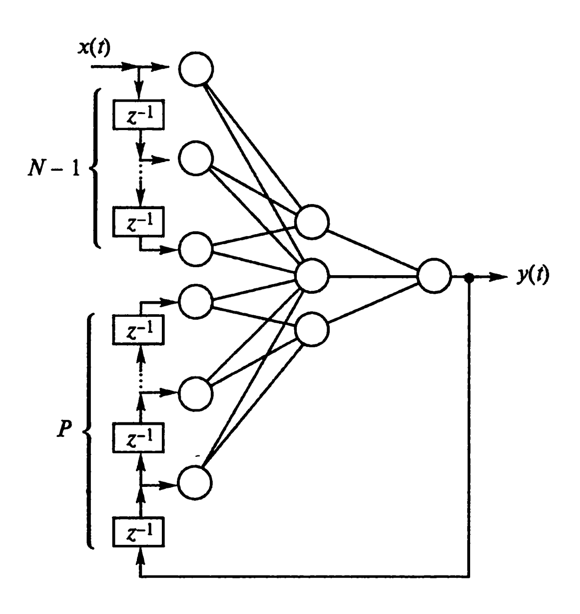


Рис. 3.29. Рекуррентная сеть, имеющая $N - 1$ элементов задержек входного сигнала и P элементов задержек в обратной связи

Если курсы валют требуется прогнозировать с учетом влияния различных астрологических или иных факторов (как это отмечалось в подразд. 3.2.7), то следует воспользоваться рекуррентной сетью с несколькими выходами. Таким свойством обладает, например, сеть Элмана [34].

Рекуррентные сети рассмотренного типа широко применяются также для математического моделирования динамических объектов. В этом случае уточнение весов выступает в роли идентификации параметров динамической математической модели. Созданная таким образом математическая модель динамического объекта может применяться для управления данным объектом — машиной, устройством, развивающимся во времени процессом.

3.5.2. Сеть Хопфилда

Хопфилд [59] обратил внимание на то, что динамический процесс, возникающий в замкнутой самой на себя рекуррентной сети может привести к некоторому устойчивому состоянию, отличающемуся от исходного. Другими словами, итерационный процесс рекуррентной сети может вывести на стационарный режим, при котором состояние сети перестанет меняться. Причем это конечное стационарное состояние сети зависит как от ее первоначального состояния, так и от значений элементов матрицы синаптических весов.

Сеть Хопфилда в классическом варианте исполнения приведена на рис. 3.30. Считается, что она не имеет входных элементов, а входной вектор задает первоначальную активность нейронов, которая затем изменяется в ходе итерационного процесса, обусловленного наличием обратных связей. В процессе итераций активность нейронов корректируется с помощью формулы

$$y_i(t+1) = \mathrm{sgn}\left(\sum_{j=1,\, i \neq j}^{N} w_{ij} y_j(t) + w_{i0} \right), \qquad (3.72)$$

т.е. принимает значения либо +1, либо –1. Согласно рис. 3.30 в схеме отсутствуют связи нейронов с их собственными выходами.

Сеть работает следующим образом. Сначала входной вектор задает начальную активность $y_i(t)$ каждого нейрона. Затем выбранный случайным образом нейрон получает взвешенные сигналы от всех остальных нейронов и обновляет свое состояние согласно формуле (3.72). Выбирается следующий нейрон, и процесс повторяется до тех пор, пока нейроны, выбранные для обновления, не перестанут изменять свое состояние. Наступает стационарный режим.

Сеть Хопфилда ведет себя подобно памяти, хранящей некоторый заданный заранее набор образов, которая пытается вспом-

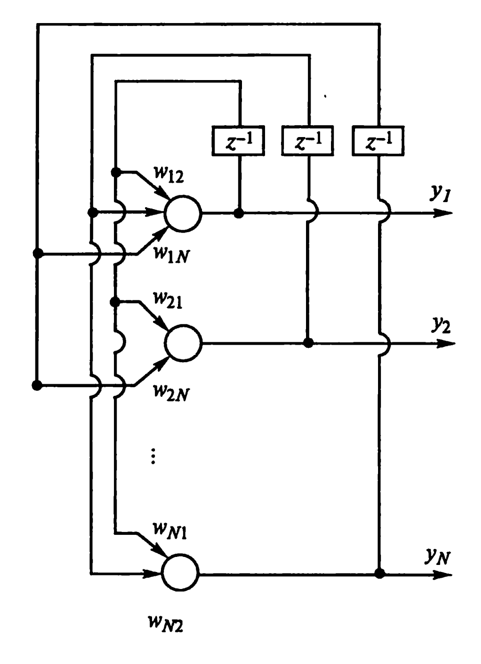


Рис. 3.30. Сеть Хопфилда

нить один из них, если ей предъявляется какой-либо из этих образов, искаженный помехами. Образы предварительно кодируются в виде векторов с бинарными компонентами. Каждый из векторов перемножается сам на себя, образуя квадратную матрицу. Затем матрицы складываются, образуя новую квадратную матрицу, главная диагональ которой обнуляется. Это и есть матрица синаптических весов w_{ij}, хранящая информацию о всех заданных образах.

Для пояснения алгоритма формирования матрицы синаптических весов приведем пример. Допустим, что предметная область содержит два образа, закодированных с помощью двух векторов: [–1, 1, –1] и [1, –1, 1]. Перемножив их самих на себя и сложив, получим квадратную матрицу:

$$\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 2 & -2 & 2 \\ -2 & 2 & -2 \\ 2 & -2 & 2 \end{bmatrix}.$$

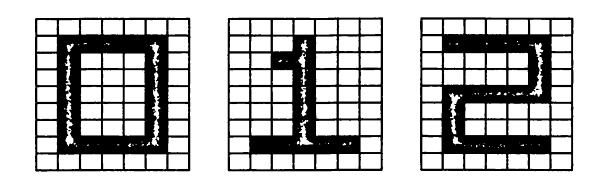


Рис. 3.31. Три образа, запомненные сетью Хопфилда

Выполнив обнуление главных диагоналей, окончательно получим

$$w_{ij} = \begin{bmatrix} 0 & -2 & 2 \\ -2 & 0 & -2 \\ 2 & -2 & 0 \end{bmatrix}.$$

Теперь предположим, что мы закодировали и ввели в сеть Хопфилда матрицу синаптических весов, соответствующую трем об-

Рис. 3.32. Предъявленный сети Хопфилда искаженный образ и его деформация за последующие 15 итераций [7]

разам, изображенным на рис. 3.31. После этого мы сформировали входной вектор, соответствующий некоторому искаженному образу, изображенному на рис. 3.32 в левом верхнем углу. Как видно из последующих кадров рис. 3.32, итерационный процесс привел к тому, что на выходе сети Хопфилда сформировался вектор, в точности соответствующий одному из ранее введенных образов.

В этом случае говорят, что входной образ *ассоциировался* с одним из введенных ранее образов и что рекуррентные сети рассмотренного типа выступают в роли *ассоциативных* запоминающих устройств.

Хопфилду математически строго удалось показать, что при любом входном векторе итерационный процесс всегда приведет к распознаванию одного из введенных ранее образов. Однако максимальное число запоминаемых сетью образов $p_{\max}$ ограничено формулой [54]:

$$p_{\max} = \frac{N}{2\ln N}, \qquad (3.73)$$

где N — число нейронов сети Хопфилда.

3.6. САМООБУЧАЮЩИЕСЯ И ГИБРИДНЫЕ СЕТИ

Рассмотренные выше нейронные сети персептронного типа обучались путем тренировки на примерах. Для обучения предоставлялась первоначальная информация о предметной области в виде набора входных векторов $\boldsymbol{X}_q$ и им соответствующих выходных векторов $\boldsymbol{D}_q$ — своего рода подсказок, с использованием которых сеть обучалась давать правильные ответы на задаваемые вопросы. Поэтому такой способ называют *обучением с учителем*.

В реальных условиях любой живой организм, взаимодействуя с окружающей средой, постоянно ощущает ее воздействие, получает своего рода подсказки, согласно которым корректирует свое поведение. Так кошка, один раз прыгнувшая на раскаленную печь, никогда больше не повторит своей ошибки. Подобных примеров можно привести тысячи, поэтому механизм обучения с учителем несомненно свойственен мозгу любого живого существа.

Однако мозг человека обладает и другими механизмами обучения. Мы в состоянии решать многие интеллектуальные задачи и без предварительного обучения. Например, мы, не задумываясь, можем выполнить кластеризацию объектов — объединить похожие между собой объекты в отдельные классы, называемые кластерами.

Рассмотрим нейронную сеть, которая без помощи учителя автоматически настраивает свои синаптические веса, решая задачу

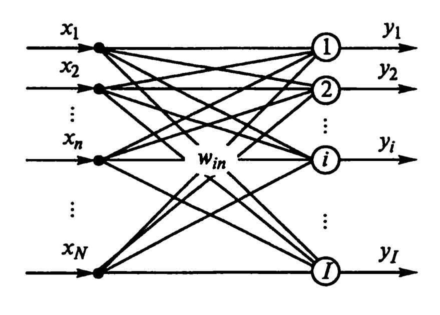


Рис. 3.33. Самообучающаяся нейронная сеть

кластеризации входных векторов. Сеть, изображенная на рис. 3.33, имеет один слой из I нейронов, каждый из которых соединен с N входами. Этот слой нейронов называют *слоем Кохонена* в честь ученого, предложившего алгоритм самообучения нейросети [61]. Веса синаптических связей каждого i-го нейрона слоя Кохонена образуют вектор синаптических связей $\boldsymbol{W}_i = [w_{i1}, w_{i2}, ..., w_{iN}]^T$, размерность которого совпадает с размерностью входных векторов $\boldsymbol{X} = [x_1, x_2, ..., x_N]^T$. Первоначально значения компонент векторов $\boldsymbol{W}_i$ задаются датчиком случайных чисел, а компоненты входного вектора $\boldsymbol{X}$ подвергаются нормализации путем деления каждой из них на длину самого вектора $\boldsymbol{X}$.

Между вектором $\boldsymbol{X}$ и каждым из векторов $\boldsymbol{W}_i$ вычисляются евклидовы расстояния

$$\|\boldsymbol{X} - \boldsymbol{W}_i\| = \sqrt{\sum_{n=1}^{N} (x_n - w_{in})^2}, \qquad (3.74)$$

среди которых выбирается наименьшее. Нейрон, у которого вектор синаптических весов $\boldsymbol{W}_i$ оказался ближе всего к входному вектору $\boldsymbol{X}$, будем называть *нейроном-победителем*, а его номер обозначим через w, т.е. нейрон-победитель имеет порядковый номер $i = w$. Синаптические веса нейрона-победителя, а также всех близлежащих к нему нейронов подвергаются корректировке по формуле Кохонена

$$\boldsymbol{W}_i(t+1) = \boldsymbol{W}_i(t) + \eta_i(t)\, G(i, w)\, (\boldsymbol{X} - \boldsymbol{W}_i(t)). \qquad (3.75)$$

В этой формуле коэффициент скорости обучения $\eta_i(t)$ уменьшается с увеличением времени t, а функция $G(i, w)$ зависит от расстояния между нейроном-победителем w и i-м нейроном сети. Обычно ее задают максимальной для $i = w$ и убывающей по мере увеличения расстояния от i-го нейрона до нейрона-победителя. Таким свойством обладает, например, функция Гаусса

$$G(i, w) = \exp\left(-\frac{\|\boldsymbol{W}_w - \boldsymbol{W}_i\|^2}{2\lambda^2}\right), \qquad (3.76)$$

в которой коэффициент λ называется *уровнем соседства*.

В другом варианте обучению по формуле Кохонена (3.75) подвергается только нейрон-победитель. В этом случае функция $G(i, w)$ имеет вид

$$G(i,w) = \begin{cases} 1 & \text{для} \quad i = w \\ 0 & \text{для} \quad i \neq w \end{cases}. \tag{3.77}$$

Этот алгоритм называется алгоритмом WTA. Название образовано начальными буквами английских слов *Winner Takes All*, что в переводе означает: «Победитель забирает все». В отличие от этого алгоритма формулу (3.76) относят к алгоритмам типа WTM — *Winner Takes Most*, что переводится как «Победитель забирает больше».

Согласно итерационной формуле Кохонена (3.75) нейрон-победитель на каждой эпохе приближает свой синаптический вектор $\boldsymbol{W}_w$ к входному вектору $\boldsymbol{X}$, как показано на рис. 3.34. В результате такого обучения каждой отдельной группе близких между собой входных векторов $\boldsymbol{X}_q$, называемой кластером, будет соответствовать один единственный нейрон, который в ходе обучения для этих векторов был победителем, причем его синаптический вектор в результате итерационного процесса (3.75) окажется в центре этого кластера.

Однако могут найтись и такие нейроны, которые ввиду их первоначальной удаленности от входных векторов так и не были ни разу победителями. Этим нейронам не будет соответствовать ни один кластер, поэтому их называют *мертвыми* нейронами. Наличие мертвых нейронов нежелательно, они снижают эффективность вычислительного алгоритма.

Проблема мертвых нейронов решается путем введения механизма, моделирующего *эффект утомления*, который известен из нейрофизиологических наблюдений. Этот эффект заключается в том, что биологические нейроны сразу после победы на некоторое время теряют свою активность и не участвуют в конкурентной борьбе.

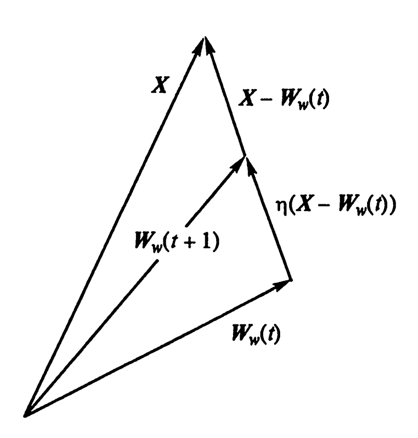


Рис. 3.34. Нейрон-победитель приближает свой синаптический вектор $\boldsymbol{W}_w$ к входному вектору $\boldsymbol{X}$

Существует несколько способов моделирования эффекта утомления нейронов, например путем введения *потенциала активности* каждого нейрона. Потенциалы активности модифицируются всякий раз после представления очередного входного вектора

$$p_i(t+1) = \begin{cases} p_i(t) - p_{\min} & \text{для} \quad i = w \\ p_i(t) + \dfrac{1}{N} & \text{для} \quad i \neq w \end{cases}. \tag{3.78}$$

В этой формуле $p_{\min}$ — минимальное значение потенциала, разрешающее нейрону участие в конкурентной борьбе. Если текущее значение потенциала какого-либо нейрона падает ниже $p_{\min}$, то этот нейрон временно отдыхает, уступая место для конкурентной борьбы другим нейронам. В результате такого обучения весовые векторы всех нейронов распределятся так, что они будут центрами различных кластеров, образованных из входных векторов, причем число кластеров, на которые разобьется входное множество векторов, будет равно числу нейронов сети.

Рассмотренная нейронная сеть представляет практический интерес, поскольку с помощью нее можно решать задачи кластеризации объектов, которые возникают, например, при необходимости сжатия информации с сохранением глобальных свойств сжимаемого множества. На рис. 3.35 приведен результат работы нейронной сети, которая разбила исходное множество на три кластера. Крестиками здесь отмечены точки исходного множества, кружочками — центры кластеров, выявленные нейросетью.

Задачи кластеризации возникают в самых разнообразных областях человеческой деятельности. Так, в педагогике часто возникает задача разделения всех учащихся на несколько классов, например, вундеркиндов, отличников, успевающих и неуспевающих. Исходной информацией для такого деления является множество различных показателей успеваемости учащихся за длительный период времени. Для решения этой задачи можно использовать слой Кохонена из четырех нейронов.

Слой Кохонена эффективно использовать в сочетании со слоями нейронов, реализующими другие нейросетевые парадигмы. Так, применение RBF-сетей, рассмотренных в подразд. 3.5, подразумевает обязательную предварительную кластеризацию входных векторов, которую удобно выполнять с помощью нейронного слоя Кохонена. На рис. 3.36 изображена гибридная сеть, содержащая слой Кохонена, выходные сигналы с которого передаются на вход обычного персептрона. Обучение гибридной нейросети осуществляется в две стадии.

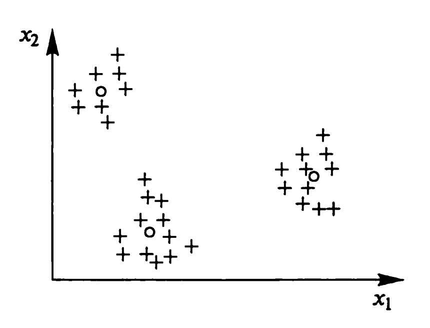


Рис. 3.35. Пример кластеризации множества

На первой стадии происходит самообучение слоя Кохонена, в

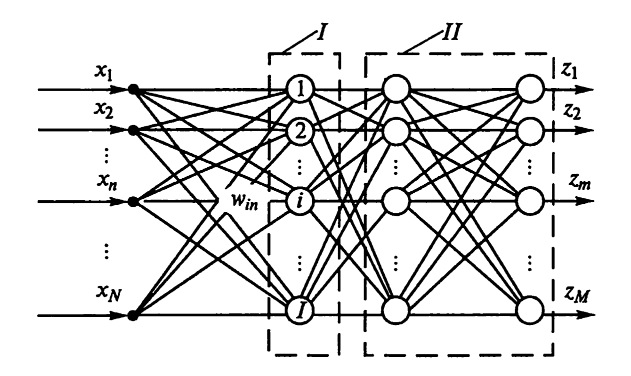


Рис. 3.36. Гибридная нейронная сеть:
I — слой нейронов Кохонена; *II* — персептронные слои

результате которого множество входных векторов разбивается на кластеры, число которых равно числу нейронов *I* слоя Кохонена, а векторы синаптических весов каждого нейрона этого слоя принимают значения, изображающиеся центрами образовавшихся кластеров. Теперь при подаче на вход сети какого-либо входного вектора каждый нейрон слоя Кохонена будет вырабатывать сумму $S_i = \sum_{n=1}^{N} w_{in} x_n$. Эти суммы нормализуются так, чтобы выходной сигнал нейрона-победителя был равен единице, а выходные сигналы остальных нейронов принимали значения в интервале (0,1). Указанная операция нормализации может осуществляться, например, с помощью формулы

$$y_i = \exp\left(-\frac{(S_{\max} - S_i)^2}{\sigma^2} \right),$$

в которой значение параметра σ подбирается индивидуально для решаемой задачи.

Персептронная часть сети обучается обычным способом, например методом обратного распространения ошибки. Обучающая выборка в этом случае состоит из выходных векторов слоя Кохонена $\boldsymbol{Y}_q$ и им соответствующих желаемых выходов сети $\boldsymbol{D}_q$. Благодаря хорошему структурированию исходных данных, выполненному слоем Кохонена, персептронная часть гибридной сети обучается во много раз быстрее, чем обычный персептрон.

В заключение отметим, что согласно современным данным, мозг человека представляет собой каскадное объединение биологических нейронных сетей различных функциональных назначений. Поэтому гибридная сеть, объединяющая слои нейронов различ-

ных нейросетевых парадигм, в большей мере соответствует современным представлениям о структуре и способе функционирования мозга. Следствием такого объединения является ее более высокая эффективность.

Контрольные вопросы

1. Постройте таблицы значимости для булевых функций «И» и «ИЛИ». Графическим способом подберите веса и пороги однонейронного персептрона, реализующего функции «И» и «ИЛИ».

2. Постройте двухслойный персептрон, реализующий функцию «Исключающее ИЛИ».

3. Составьте программу обучения однонейронного персептрона с помощью правил Хебба и дельта-правила.

4. С помощью составленной программы попытайтесь обучить однонейронный персептрон логическим операциям «И», «ИЛИ», «Исключающее ИЛИ».

5. Спроектируйте и обучите нейросеть прогнозированию курса американского доллара по отношению к российскому рублю.

А. Возьмите из сети Internet и изобразите графически данные по изменению курса доллара за последние три месяца.

Б. По данным двух первых месяцев методом окон обучите нейросеть прогнозированию курса доллара на один (или более) день вперед.

В. Определите среднеквадратичную ошибку прогноза, используя в качестве тестовых примеров данные последнего месяца.

Г. Введите в нейросеть дополнительный входной нейрон, в котором закодируйте день недели прогнозируемого дня. Повторите пункты Б—В.

Д. В дополнительном нейроне сети закодируйте данные о солнечной активности, взятые из сети Internet. Повторите пункты Б—В.

Е. В дополнительном нейроне сети закодируйте сведения о фазах Луны и повторите пункты Б—В.

Ж. Сравните среднеквадратичные ошибки прогноза на тестовых выборках, сделайте заключение о степени влияния на курс доллара исследованных факторов.

6. Нарисуйте схему RBF-сети с минимальным числом нейронов, способную моделировать функции «И», «ИЛИ», «Исключающее ИЛИ».

ГЛАВА 4
РАСПОЗНАВАНИЕ ОБРАЗОВ

4.1. ПРОБЛЕМА РАСПОЗНАВАНИЯ ОБРАЗОВ

Человеческий мозг, так же как и мозг животных, с самого своего рождения и на протяжении всей жизни ежеминутно решает задачи распознавания образов. Ребенок или детеныш животного с первых минут своего появления на свет узнает пищу, мать, ее голос, окружающие предметы. По мере взросления ребенок учится узнавать свои игрушки, комнату, дом, множество необходимых предметов, лица друзей, их речь, музыку, буквы, слова, книги и т.д.

В своей повседневной жизни человек настолько легко справляется с задачами распознавания, что это считается само собой разумеющимся. Между тем, попытки моделирования на компьютерах этих высокоинтеллектуальных функций наталкиваются на весьма серьезные трудности.

В настоящее время наибольших успехов удалось добиться в распознавании зрительных образов, таких как печатные символы. Не вызывает сомнений полезность известных программ распознавания текстовой информации — FineReader и CuneiForm. Функции обнаружения и распознавания военных объектов противника уже давно закладываются в бортовые компьютеры ракет, самолетов, кораблей и подводных лодок.

Какие идеи и принципы могут быть заложены в основу распознающих систем? Первое, что приходит в голову, — действовать «с позиции грубой силы»: заложить в компьютер как можно больше известных образов-шаблонов и сравнивать их с поступающими для распознавания неизвестными образами. Однако этот путь сразу заводит в тупик. Предположим, что зрительное изображение считывается с помощью стандартной системы светочувствительных элементов — 32 позиции по ширине и 48 по высоте, т.е. всего 1536 элементов. Но даже на такой грубой сетке можно воспринять порядка 10^{460} возможных образов. Налицо комбинаторный взрыв. Хранить в памяти такое число шаблонных изображений и осуществлять с ними сравнение поступающих на вход образов невозможно.

Поэтому на практике системы распознавания на первой стадии обязательно обрабатывают изображение и выделяют харак-

терные признаки, качественные или количественные. Таким образом, количество информации для распознавания существенно уменьшается.

Следующая идея, которая обычно используется в распознающих системах, — это идея обучения. Она является обязательным элементом многих современных интеллектуальных систем. Мы рассмотрим ее реализацию на примерах первых интеллектуальных систем, ставших классическими, — пандемониуме Селфриджа и персептроне Розенблатта.

4.2. ПАНДЕМОНИУМ СЕЛФРИДЖА

Как уже отмечалось, системы распознавания строятся таким образом, что на первом этапе поступающие на вход изображения обрабатываются с целью выделения наиболее существенных признаков. Эти признаки могут представлять собой определенные участки изображения либо иметь достаточно общий характер, например давать ответ на конкретный вопрос: «Имеется ли на изображении вертикальная черта?».

Система распознавания образов, предложенная Оливером Селфриджем [66] и называемая пандемониумом, состоит из элементов — демонов. Демоны — это относительно автономные сущности, выполняющие элементарные функции. На самом нижнем уровне находятся демоны данных, или демоны изображения (рис. 4.1), которые играют роль светочувствительных элементов сетчатки глаза.

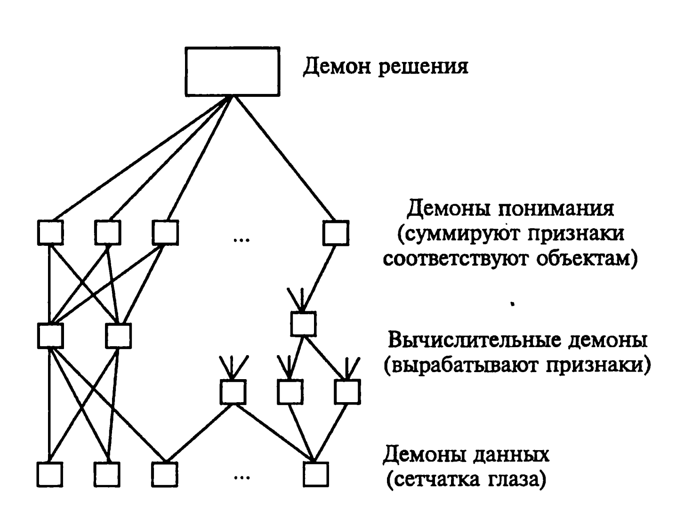


Рис. 4.1. Пандемониум Селфриджа

На самом верхнем уровне находится демон решения, который определяет выход всей системы в целом — выдает сообщение, к какой категории относится узнанный образ. Ниже демона решения имеется некоторое число демонов понимания, каждый из которых соответствует одной из узнанных категорий. Например, первый демон понимания соответствует кошке, второй — собаке, третий — человеку и т.д. Вычислительные демоны, обрабатывая визуальную информацию от демонов данных, вырабатывают признаки и передают их демонам понимания.

Идея пондемониума состоит в том, что каждый демон понимания должен определить меру соответствия поступающих на его вход признаков категории образа, представленного этим демоном понимания. Чем лучше это соответствие, тем более сильный сигнал посылается к демону решения, который сравнивает интенсивность сигналов и выбирает наиболее сильный.

О. Селфридж так образно описал работу предлагаемого устройства [66]: «Задача демона понимания состоит в том, чтобы исследовать поступающие признаки и выкрикивать название своего класса, если он считает, что объект относится именно к этому классу. Демон должен кричать громко, если он уверен в своем решении, и тихо, если не уверен. Однако общий шум, производимый демоном понимания, зависит не только от его стараний, но и от способности кричать. Последнее определяет всемогущий демон решения, который наделяет демонов первого порядка сильными или слабыми голосами. Таким образом, каждый демон понимания выкрикивает название своего класса с интенсивностью, зависящей от его собственных оценок и от силы данного ему голоса. Демон решения, который ведет себя как председатель собрания, где проводится голосование, решает, название какого класса было выкрикнуто громче всех».

Идея, предложенная О. Селфриджем, программируется весьма просто. Демоны понимания всего лишь вычисляют взвешенные суммы сигналов, поступающих от вычислительных демонов. Так, i-й демон понимания вычисляет свой выходной сигнал D_i следующим образом:

$$D_i = \sum_j w_{ij} d_j, \tag{4.1}$$

где w_{ij} — весовые множители, устанавливающие значимость признаков, поступающих для суммирования; d_j — выходной сигнал j-го вычислительного демона.

Суммирование ведется по всем вычислительным демонам. Фактически демоны понимания различаются между собой только значениями указаных весовых множителей.

Приведем простой пример. Пусть сущностью первого демона понимания является кошка, второго — собака, третьего — чело-

Таблица 4.1

Весовые множители, устанавливающие значимость признаков для объектов распознавания

Демон понимания	Наличие усов	Наличие шерсти	Наличие хвоста	Вес более 50 кг
Кошка	$w_{11} = 10$	$w_{12} = 10$	$w_{13} = 10$	$w_{14} = 0$
Собака	$w_{21} = 0$	$w_{22} = 10$	$w_{23} = 8$	$w_{24} = 0$
Человек	$w_{31} = 2$	$w_{32} = 0$	$w_{33} = 0$	$w_{34} = 10$

век. Зададим для них по десятибалльной шкале значения весовых множителей, которые поместим в табл. 4.1.

Допустим, что на считывающем устройстве пандемониума появляется образ в виде кошки. Вычислительные демоны сформируют следующие признаки: $d_1 = 1$ (что означает — есть усы), $d_2 = 1$ (есть шерсть), $d_3 = 1$ (имеется хвост), $d_4 = 0$ (вес не более 50 кг).

Демоны понимания произведут обработку признаков:

демон кошки — $D_1 = 10 \times 1 + 10 \times 1 + 10 \times 1 + 0 \times 0 = 30$;

демон собаки — $D_2 = 0 \times 1 + 10 \times 1 + 8 \times 1 + 0 \times 0 = 18$;

демон человека — $D_3 = 2 \times 1 + 0 \times 1 + 0 \times 1 + 10 \times 0 = 2$.

Таким образом, наибольшее число баллов набрала кошка.

Теперь предположим, что на входе появился человек. Вычислительные демоны дадут признаки: $d_1 = 0$ (что означает — нет усов), $d_2 = 0$ (нет шерсти), $d_3 = 0$ (нет хвоста), $d_4 = 1$ (вес более 50 кг).

Демоны понимания произведут обработку поступивших признаков:

демон кошки — $D_1 = 10 \times 0 + 10 \times 0 + 10 \times 0 + 0 \times 1 = 0$;

демон собаки — $D_2 = 0 \times 0 + 10 \times 0 + 8 \times 0 + 0 \times 1 = 0$;

демон человека — $D_3 = 2 \times 0 + 0 \times 0 + 0 \times 0 + 10 \times 1 = 10$.

Вывод очевиден — на входе человек.

Пандемониум представляет собой обучающееся устройство, и каждый демон понимания осуществляет настройку своего способа комбинации выходных сигналов вычислительных демонов. Эта подстройка выполняется путем подбора весов w_{ij} и определяется обратной связью с окружающей средой, указывающей на правильность или полезность принимаемого решения, т.е. здесь подразумевается присутствие учителя, который сообщает системе правильную классификацию. Конкретные алгоритмы настройки системы могут быть самыми разнообразными и включать в себя математические методы оптимизации.

Когда весовые коэффициенты более-менее подобраны и принимаемое решение близко к оптимальному, то для любого вычислительного демона становится возможным вычислить его ценность для всей системы в целом. Ценность вычислительного демо-

на определяется тем, насколько используется его выход. Мерой такой ценности может быть величина

$$W_i = \sum_i |w_{ij}|. \tag{4.2}$$

Определение ценности позволяет производить изменения в используемом множестве вычислительных демонов. Например, можно автоматически исключать малоценные демоны и заменять их другими. Таким образом система приобретает самоорганизующийся характер и ее настройка не сводится просто к самооптимизации параметров.

Автором системы были предложены два способа получения новых вычислительных демонов. Оба они основаны на том соображении, что целесообразно создавать демоны, имеющие что-то общее с уже существующими, которые доказали свою ценность. Эти методы называются слиянием и делением с мутацией.

Слияние заключается в том, что выходные сигналы двух демонов высокой ценности комбинируются между собой, например по принципу «все или ничего». На рис. 4.1 на вход одного из вычислительных демонов поступают сигналы от двух других демонов. Этот результирующий демон возник в результате слияния.

4.3. ПЕРСЕПТРОН РОЗЕНБЛАТТА

Способ распознавания, заложенный в пандемониуме О. Селфриджа, плохо согласуется с нашими представлениями о процессах, происходящих в мозге. Поэтому этот способ характерен для кибернетики «черного ящика». Альтернативным подходом является попытка копирования процессов коры головного мозга, реализованная в другом устройстве распознавания образа — персептроне.

Термин «персептрон» был введен в 1950-х гг. Фрэнком Розенблаттом для некоторого класса интеллектуальных систем распознавания образов, способных обучаться на опыте. Идея персептрона и примеры его применения для распознавания цифр и букв рассмотрены в гл. 3. Здесь же мы остановимся на принципе действия персептрона в его первоначальном исполнении исходя из аналогий с пандемониумом Селфриджа.

Первоначально персептрон Розенблатта (рис. 4.2) содержал узлы трех типов. Сенсорные, или с-узлы, имитировали светочувствительные клетки сетчатки глаза. Они соответствовали демонам изображения, или демонам данных, пандемониума Селфриджа. Обычно предполагается, что с-узлы являются элементами типа «все или ничего», но это не обязательно.

Следующий слой состоял из ассоциативных, или а-узлов, которые в общих чертах соответствуют вычислительным демонам

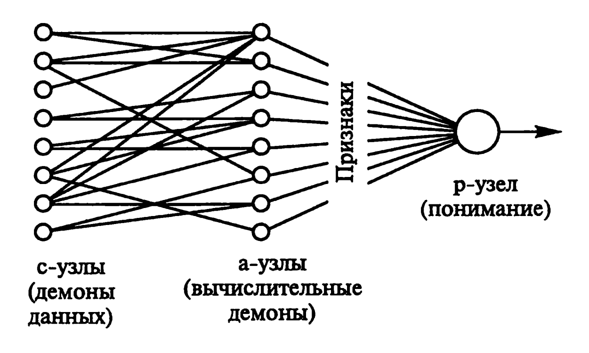


Рис. 4.2. Персептрон Розенблатта

пандемониума. В первоначальных вариантах исполнения персептрона соединения, идущие от с-узлов, формировались случайным образом еще в процессе конструирования системы, поэтому они определяли некоторые случайные свойства изображения. Как и в пандемониуме, при обучении персептрона вычислялись данные о ценности каждого а-узла. Входные соединения а-узла, ценность которых в процессе работы оказывалась малой, аннулировались, после чего случайным или псевдослучайным образом устанавливался новый набор соединений.

Выходы а-узлов были соединены с узлами реакции, или р-узлами, соответствующими демонам понимания пандемониума. В отличие от пандемониума р-узел дает только ответ «да» или «нет».

Как а-узлы, так и р-узлы персептрона представляли собой математические нейроны, алгоритм действия которых описан в гл. 3. Некоторые из соединений между узлами являлись возбуждающими, а некоторые — тормозящими. Веса синапсов, идущих к р-узлам, изменялись в процессе обучения персептрона.

Алгоритм обучения персептрона состоял в следующем. Если реакция р-узла являлась правильной (т.е. он срабатывал, когда образ принадлежал к распознаваемому классу, или не срабатывал, когда образ не принадлежал указанному классу), то веса не изменялись.

Если р-узел не срабатывал, когда распознаваемый образ на самом деле относился к рассматриваемому классу, то веса синапсов, бывших активными, увеличиваются на некоторую величину *c*. С другой стороны, если р-узел срабатывал на образ, который не принадлежал распознаваемому классу, то веса активных синапсов уменьшались на величину *c*.

С помощью этих весов вычислялись ценности каждого а-узла, и если ценность была мала, то его синапсы (связи с с-узлами) разрушались и строились новые.

Как уже отмечалось ранее, возможности персептрона были всесторонне исследованы математически. В частности, была до-

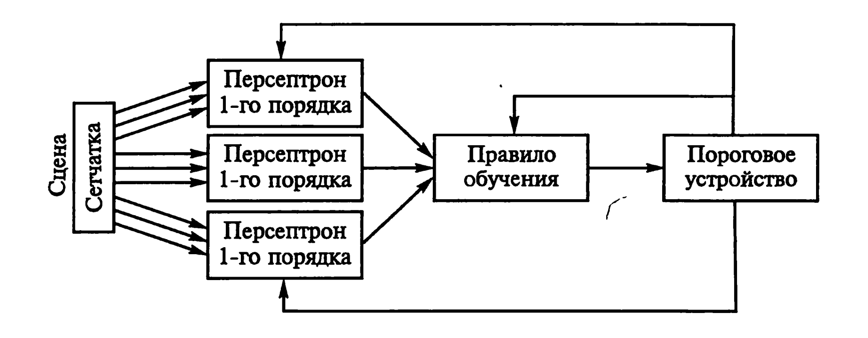


Рис. 4.3. Многослойный персептрон Гамбы

казана теорема сходимости, формулировка которой приведена в подразд. 3.1.3.

На рис. 4.3 приведена схема популярного в свое время многослойного персептрона Гамбы, являющегося развитием изложенного выше однослойного персептрона.

В заключение отметим, что несмотря на отмеченное сходство идей, персептрон в большей степени, чем пандемониум, соответствует нашим представлениям о структуре и процессах функционирования мозга. Возможно поэтому он оказался более эффективен и получил в наши дни дальнейшее развитие и применение в нейросетевых и нейрокомпьютерных технологиях.

4.4. РАСПОЗНАВАНИЕ СИМВОЛОВ

4.4.1. Методы распознавания символов

Распознавание символов по их графическому представлению — одна из самых старых и традиционных задач искусственного интеллекта. Еще в 1960—1970-е гг. были написаны десятки диссертаций и предложены сотни методов решения этой проблемы. Однако только сейчас, когда сканеры и компьютеры стали общедоступны, распознавание символов получило полноценное практическое применение.

Все существующие в настоящее время методы распознавания символов можно разделить на три вида: шаблонный (эталонный), структурный и признаковый.

Шаблонный метод. В большинстве систем шрифт, подлежащий распознаванию, хорошо известен и распознавание является лишь вопросом использования эталонов. В этом случае просто нужно ввести некоторый допуск на расхождение между символом и шаблоном с учетом дефектов печати и помарок на бумаге.

Можно вычислить меру соответствия между распознаваемым изображением и эталоном, хранящимся в памяти компьютера. Такой мерой может быть, например, доля общей площади изображения и эталона при наложении их друг на друга.

Структурный метод. Распознаваемый объект описывается как граф, узлами которого являются элементы входного объекта, а дугами — пространственные отношения между ними. Системы, реализующие подобный подход, обычно работают с векторными изображениями.

Структурными элементами являются линии, составляющие символ. Так, для буквы «р» — это вертикальный отрезок и дуга.

Признаковый метод. Согласно этому методу изображение каждого распознаваемого символа представляется как объект в n-мерном пространстве признаков. Сами признаки устанавливаются и вычисляются на стадии предварительной обработки изображений. Полученный n-мерный вектор сравнивается с эталонными, и изображение относится к наиболее подходящему из них.

Каждый искусно подобранный признак резко сокращает число возможных букв. Например, достаточно знать, что левый верхний угол буквы скруглен, и из тридцати трех букв русского алфавита остаются лишь девять кандидатов (а, б, е, з, о, с, ф, э, я). Букв, содержащих две «ноги» (вертикальные отрезки на всю высоту буквы) всего десять (и, й, л, м, н, п, ц, ш, щ, ы). Таким образом, задав несколько простых вопросов, можно по ответам на них однозначно определить букву. Как видим, это тот самый подход, который заложен в пандемониуме и персептроне.

Все методы имеют свои недостатки, и, разумеется, лучше применять их комбинации. Теоретически это просто. Однако программы, позволяющие подойти к практическому решению этой задачи, были созданы только в конце 1980-х гг.

Сегодня системы распознавания текстов составляют важную часть большинства технологий хранения и обработки документов. Особенно успешно они используются студентами при выполнении курсовых и дипломных работ.

4.4.2. Предварительная обработка изображений

На первом этапе обработки изображений, поступающих со считывающего устройства, решается задача фильтрации, т.е. понижения разного рода шумов (помех), вносимых измерительными системами и каналами связи.

Допустим, что в компьютер поступает некоторое двумерное изображение, которое можно описать функцией $f(x, y)$, представляющей собой распределение, например, яркости, светимости, плотности. В дальнейшем эту функцию будем называть *зачерненностью.*

Для последующей обработки необходимо выполнить квантование (дискретизацию) этой функции как по пространству, так и по значению зачерненности.

Сегментация. Обычно изображение состоит из двух частей: компонентов образа, подлежащего распознаванию, и фона. Под сегментацией понимается отнесение элементов изображения либо к компонентам образа, либо к фону. Существуют два метода сегментации — разделение по порогу и обнаружение края.

Разделение по порогу. Сегментация осуществляется исключительно на основе значения функции зачерненности каждого элемента изображения. Если функция $f(x, y) > \theta$, где θ — значение некоторой пороговой величины, то соответствующий элемент изображения классифицируется как компонента распознаваемого образа, в противном случае он относится к фону.

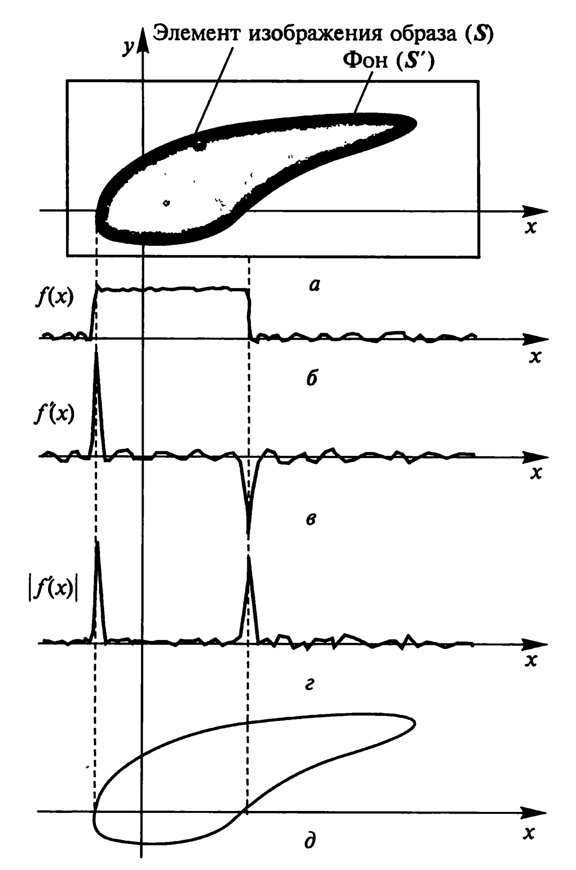


Рис. 4.4. Распределение функции зачерненности $y = f(x, y)$ в плоскости $x — y$ (*а*), в сечении $y = 0$ (*б*), распределение ее производной по x (*в*), модуля производной (*г*) и особенностей поля производной (*д*)

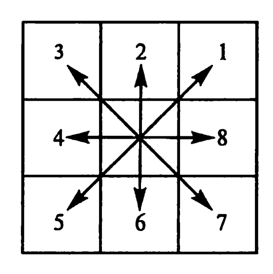


Рис. 4.5. Правило формирования цепного кода

На сегментированном таким образом изображении легко выделить контуры распознаваемого образа. Обозначим через S некоторое интересующее нас подмножество элементов изображения, относящееся к компонентам образа, а через S' — дополнение подмножества (фон). Тогда контур будет состоять из элементов подмножества S, которые имеют в качестве соседних элементы подмножества S'. Пользуясь этим правилом, несложно создать алгоритм, выявляющий контурные пикселы.

Обнаружение края. Обычно граница между компонентами образа и фоном характеризуется резким изменением функции зачерненности. Поэтому обнаружить эту границу можно путем дифференцирования функции $f(x, y)$ по координатам, как схематично показано на рис. 4.4. Ложные края можно удалить, а утраченные — восстановить, используя априорную информацию об образе.

Обработка сегментированных изображений. Топологические особенности контура можно выявить, пользуясь цепным кодом. Правило формирования цепного кода иллюстрирует рис. 4.5. Для кодирования очередного пиксела контура используются числа от 1 до 8 в зависимости от его расположения относительно начального пиксела.

По записи контура в цепном коде можно вычислить ряд признаков распознаваемого образа, в частности, площадь, ограниченную контуром (если он замкнут), кривизну в определенном пикселе контура, а также определить, замкнут ли контур. Можно установить, обладает ли контур локальной вогнутостью или выпуклостью.

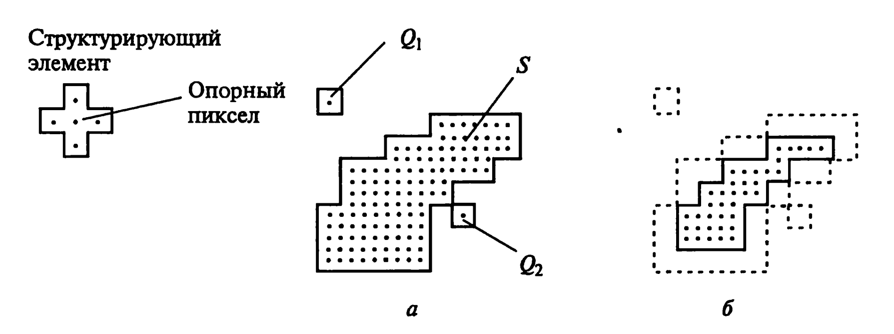
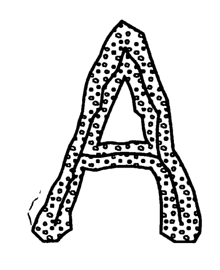


Рис. 4.6. Распознаваемый образ S и случайные частицы Q_1 и Q_2 до (*а*) и после (*б*) применения операции поверхностного разрушения

Рис. 4.7. Изображение буквы и ее остова



С выявленным компонентом образа можно производить операции *поверхностного разрушения* (сжатия, прореживания) и *наращивания* (расширения). Эти операции применяют для улучшения качества изображения. Для реализации указанных операций необходимо перемещать некоторый структурирующий элемент по изображению образа. Если структурирующий элемент полностью укладывается в подмножество *S*, перемещение структурирующего элемента продолжается дальше без модификации изображения. Если же структурирующий элемент не входит в *S* полностью, то элемент изображения, соответствующий положению опорного пиксела структурирующего элемента, удаляется. На рис. 4.6 приведен пример этой операции. Структурирующий элемент (см. рис. 4.6, *а*) состоит из центрального пиксела, являющегося опорным, и четырех соседних с ним пикселов. На рис. 4.6, *б* удаленные после операции поверхностного разрушения пикселы изображены штриховыми линиями.

Операция, обратная поверхностному разрушению, называется наращиванием. Эту операцию можно также трактовать как поверхностное разрушение подмножества *S'*.

В случае применения операции наращивания к подмножеству *S* компоненты изображения расширяются — к ним добавляются пикселы фона, соседние с *S*.

Операции разрушения и наращивания можно применять многократно. Таким способом можно добиться очищения изображений — удаления малых случайных частиц (таких как Q_1 и Q_2 на рис. 4.6, *а*) и заполнения случайных промежутков.

Многократное поверхностное разрушение контура в конечном итоге приводит к полному устранению подмножества *S*. Однако с помощью этой же операции можно составить алгоритм, выделяющий *остов* изображения — линии толщиной в один пиксел, которые проходят посередине подмножества *S*, сохраняя его топологию. Пример выделения остова приведен на рис. 4.7.

Остовы выявляют структуру изображения, и их можно использовать при формулировке признаков.

4.4.3. Распознавание по методу Паркса

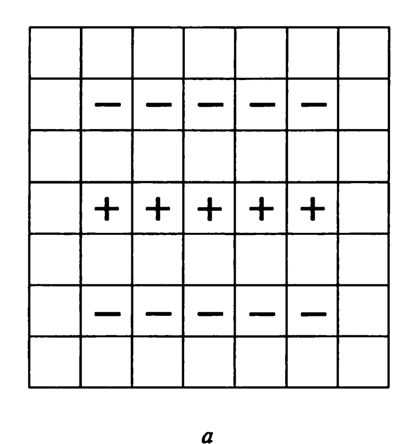
Джоном Парксом была предложена оригинальная система выделения признаков, полезных для распознавания символов, главным образом, букв и цифр. Система Паркса работает с остовами, оставшимися после применения процедур сегментации и поверх-

ностного разрушения (см. подразд. 4.4.2). Входное изображение просматривается с целью обнаружения отрезков прямых, которые могут быть ориентированы различным образом. Просмотр осуществляется средствами электроники, но точно так же, как если бы на каждый участок исходного массива, содержащий 7 × 7 пикселов, накладывалась решетка, состоящая из 7 × 7 фотоэлементов. Эта решетка движется по изображению слева направо, перемещаясь каждый раз на один пиксел. Пройдя таким образом строку, решетка опускается на один пиксел вниз и снова перемещается слева направо. В каждом положении регистрируется выходной сигнал каждого фотоэлемента и производятся определенные сравнения между группами фотоэлементов.

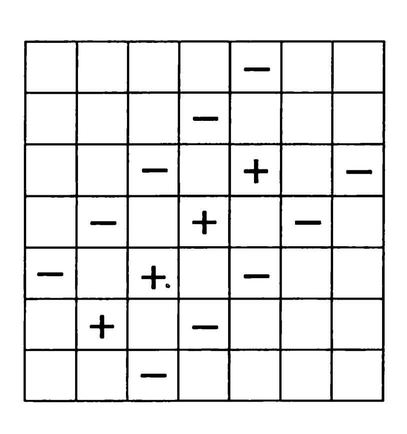
На рис. 4.8 показано, какие сравнения могут осуществляться между подмножествами фотоэлементов. Среднее из значений фотоэлементов, помеченных знаком «+», сопоставляется со средним из значений, помеченных знаком «–». Если оказывается, что изображение более темное под ячейками «+», то мы получаем данные относительно возможности присутствия в этой области горизонтального отрезка. Строки фотоэлементов взяты длиной всего в пять пикселов, что позволяет не слишком строго отбирать линии по направлению.

Аналогичное сопоставление данных, полученных от фотоэлементов, составляющих вертикаль, выделяет отрезки линий, расположенных примерно вертикально. По такому же принципу выделяются и наклонные линии.

Таким образом, исходное множество пикселов, в которых была указана яркость, превращается в массив ячеек, несущих информацию о наличии отрезка прямой, идущего в том или ином направлении. Так осуществляется переход к признакам.



а



б

Рис. 4.8. Расположение пикселов при поиске горизонтальных (*а*) и наклонных (*б*) отрезков изображения

Новый массив опять рассматривается с помощью ячейки 7 × 7 (рис. 4.9), которая при определенных условиях указывает на наличие некоторого морфологического признака символа. Таким признаком может быть изменение направления линии; острые, прямые или тупые углы; слияние или пересечение линий. Всего получается 54 различных морфологических признака.

Представление символа сводится к перечислению его морфологических признаков с указанием их примерного расположения. Простая программа, составленная на основе статистических данных, собранных на свободно написанных цифрах, позволяет осуществлять довольно надежное распознавание.

						D
					D	*D*
	H					
H	*H*	*H*	*H*			
					d	
		V				*d*

Рис. 4.9. Пример заполнения решетки на втором этапе сканирования:

H — наличие горизонтального участка; *V* — наличие вертикального участка; *D* и *d* — наличие наклонных участков

Система выделения признаков, предложенная Парксом (первоначальный просмотр изображения с помощью решетки и обнаружение отрезков прямых) очень хорошо соответствует процессу, который происходит в зрительной коре мозга животных на начальных этапах обработки изображений. Поэтому метод Паркса можно классифицировать как нейрокибернетический подход.

4.4.4. Современные системы распознавания текстов

В настоящее время получили распространение две конкурирующие между собой системы распознавания текстовой информации — FineReader (компания «Бит») и CuneiForm (компания «Cognitive Technologies»). Разработчики этих систем тщательно оберегают свои ноу-хау, однако некоторые принципы, используемые этими программами, становятся известными.

Основные принципы работы системы FineReader. Компанией «Бит» была разработана специальная технология распознавания символов, которая получила название фонтанного преобразования (от англ. font — шрифт), а на ее основе — коммерческий продукт — система оптического распознавания FineReader.

В основе фонтанного преобразования лежит так называемый принцип целостности. В соответствии с ним любой воспринимаемый объект рассматривается как целое, состоящее из частей, связанных между собой определенными отношениями.

Для выделения целого требуется определить его части. Части же, в свою очередь, можно рассматривать только в составе целого. Поэтому целостный процесс восприятия может происходить только в рамках гипотезы о воспринимаемом объекте — целом.

При распознавании текста человеком любое решение принимается путем последовательного выдвижения и проверки гипотез с привлечением знаний как о самом исследуемом объекте, так и о содержании всего текста.

Первый шаг восприятия — это формирование гипотезы о воспринимаемом объекте. Гипотеза может формироваться как на основе априорной модели объекта, контекста и результатов проверки предыдущих гипотез (процесс «сверху вниз»), так и на основе предварительного анализа объекта («снизу вверх»). Второй шаг — уточнение восприятия (проверка гипотезы), при котором производится дополнительный анализ объекта в рамках выдвинутой гипотезы и в полную силу привлекается контекст.

Каждая гипотеза должна быть объектом, который можно было бы оценить или сравнить с другими. Гипотезы выдвигаются последовательно, а затем объединяются в список и сортируются на основе предварительной оценки. Для окончательного выбора гипотезы активно используются контекст и другие дополнительные источники знания.

Как уже отмечалось выше, сегодня известны три подхода к распознаванию символов — шаблонный, структурный и признаковый. Принципу целостности отвечают лишь первые два. Поэтому компания «Бит» принципиально не использует признаковый подход. Вместе с тем разработчики FineReader обращают внимание на недостатки шаблонного и структурного подходов.

Шаблонные системы довольно устойчивы к дефектам изображения и имеют высокую скорость обработки входных данных, но надежно распознают только те шрифты, шаблоны которых им известны. Если распознаваемый шрифт хоть немного отличается от эталонного, шаблонные системы могут делать ошибки даже при обработке очень качественных изображений.

К недостаткам структурных систем относится их высокая чувствительность к дефектам изображения, нарушающим составляющие элементы. Кроме того, векторизация изображений сама по себе может добавлять дополнительные дефекты.

Фонтанное преобразование, предлагаемое разработчиками системы FineReader, по их мнению, совмещает в себе достоинства шаблонной и структурной систем и позволяет избежать недостатков, присущих каждой из них по отдельности.

В основе этой технологии лежит использование структурно-пятенного эталона (термин введен разработчиками системы). Изображение представляется в виде набора пятен, связанных между собой n-арными отношениями, задающими структуру символа. Эти

отношения, т.е. расположение пятен друг относительно друга, образуют структурные элементы, составляющие символ. Так, отрезок — это один тип *n*-арных отношений между пятнами, эллипс — другой, дуга — третий.

Другие отношения задают пространственное расположение образующих символ элементов. Используются связи между структурными элементами, которые определяются либо метрическими характеристиками этих элементов (например, «длина больше»), либо их взаимным расположением на изображении (например, «правее», «соприкасается»).

При задании структурных элементов и отношений применяются конкретизирующие параметры, позволяющие доопределить структурный элемент или отношение при использовании этого элемента в эталоне конкретного класса. Конкретизирующими могут являться, например, параметры, задающие диапазон допустимой ориентации отрезка или предельное допустимое расстояние между характерными точками структурных элементов. Пример задания конкретизирующих параметров для буквы «А» приведен на рис. 4.10.

Конкретизирующие параметры используются также для определения качества распознавания. Так, если вычисленный конкретизирующий параметр находится где-то посередине допустимого интервала, то качество распознавания считается высоким, в противном случае — низким. Таким образом, система сама себе ставит оценку.

Построение и тестирование структурно-пятенных эталонов для классов распознаваемых объектов — процесс сложный и трудоемкий. База изображений, которая используется для отладки описаний, должна содержать примеры хороших и плохих (предельно допустимых) изображений для каждой графемы, а изображения базы разделяются на обучающее и контрольное множества.

Разработчик описания предварительно задает набор структурных элементов (разбиение на пятна) и отношения между ними.

Система обучения по базе изображений автоматически вычисляет параметры элементов и отношений. Полученный эталон проверяется и корректируется по контрольной выборке изображений данной графемы. По этой же выборке проверяется результат распознавания, т.е. оценивается качество подтверждения гипотез.

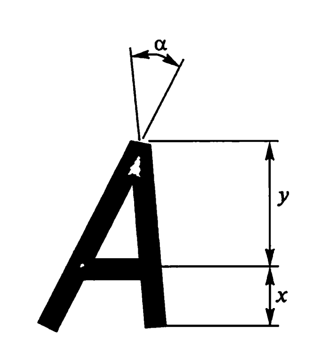


Рис. 4.10. Пример задания конкретизирующих параметров:
$\alpha_0 < \alpha < \alpha_1$; $k_0 < \frac{x}{y} < k_1$

Распознавание с использованием структурно-пятенного эталона происходит следующим образом. Эталон накладывается на изображение, и отношения между выделенными на изображении пятнами сравниваются с отношениями пятен на эталоне. Если выделенные на изображении пятна и отношения между ними удовлетворяют эталону некоторого символа, то данный символ добавляется в список гипотез о результате распознавания входного изображения. Гипотезы сортируются согласно оценкам их качества, и преимущество отдается той, которая имеет максимальный балл.

Основные принципы работы системы CuneiForm. По мнению разработчиков CuneiForm, их система является более интеллектуальной, поскольку задача распознавания текста в ней решается на основе взаимодействия структурного, признакового, растрового, дифференциального и лингвистического уровней.

Работает система по принципу «одной кнопки». Это означает, что при нажатии кнопки «Сканируй и распознавай» запускается весь процесс обработки документа: сканирование, фрагментация страницы на текстовые и графические блоки, сегментация, сжатие — расширение, выделение остовов, распознавание текста, проверка орфографии и формирование выходного файла.

Интеллектуальный алгоритм позволяет автоматически подобрать оптимальный уровень яркости сканера (адаптивное сканирование) в зависимости от фона документа, сохранить иллюстрации или, в зависимости от решаемой задачи, удалить ненужные графические элементы для максимального сокращения последующего редактирования.

За распознавание текста отвечает целый ряд модулей (сканирования, выбора яркости, предобработки документа, фрагментации и др.), каждый из которых решает свою задачу. На вход модуля распознавания поступает полученное после сканирования изображение.

В CuneiForm используется несколько методов распознавания. Образ каждого символа раскладывается на отдельные элементы — события. К примеру, событием является фрагмент от одной линии пересечения до другой. Совокупность событий представляет собой компактное описание символа.

Другие методы основаны на использовании соотношения «масс» отдельных элементов символов и описании их характерных признаков (закругления, прямые углы и т.д.). По каждому из этих описаний существуют базы данных, в которых находятся соответствующие эталоны. Поступающий на обработку элемент изображения сравнивается с эталоном, а затем на основании этого сравнения решающая функция выносит вердикт о соответствии изображения конкретному символу.

Таким образом, в отличие от FineReader, здесь в полной мере используется признаковый подход, в частности, рассмотренный в подразд. 4.4.3 метод Паркса.

Поскольку в системе CuneiForm используется не один, а сразу несколько методов распознавания, распознаваемый образ сравнивается не с одним, а с несколькими типами эталонов, представленными различными способами.

Кроме того, существуют алгоритмы, которые позволяют работать с текстами низкого качества. Так, для разрезания «склеенных» символов существует метод оценки оптимальных разбиений (ноу-хау не раскрываются). И, напротив, для «рассыпанных» элементов разработан алгоритм их соединения.

В версии CuneiForm 96 впервые применен алгоритм самообучения. Принцип его состоит в следующем. В каждом тексте присутствуют четко и нечетко пропечатанные символы. Если после того, как система распознала текст (как это делает обычная система, например предыдущая версия CuneiForm 2.95), выясняется, что точность оказалась ниже пороговой, то производится дораспознавание текста на основе шрифта, который генерируется системой по хорошо пропечатанным символам.

Как утверждают разработчики, результаты применения CuneiForm 96 показали, что использование самообучающихся алгоритмов позволяет повысить точность распознавания низкокачественных текстов в 4—5 раз. Но главное преимущество заключается в том, что самообучающиеся системы обладают гораздо бóльшим потенциалом повышения точности распознавания, открывая новое направление в теории распознавания символьной информации.

Не ограничиваясь геометрическими методами распознавания, разработчики системы CuneiForm дополнили ее орфографическим, синтаксическим и семантическим дораспознаваниями и контролем. При этом разработчикам пришлось решить две важные задачи. Во-первых, было необходимо реализовать быстрый доступ к большому (порядка 100 000 слов) словарю. В результате удалось построить систему хранения слов, где на каждое слово уходило не более одного байта, а доступ осуществлялся за минимальное время (ноу–хау не раскрывается). Во-вторых, потребовалось построить систему коррекции результатов распознавания, ориентированную на альтернативность событий (подобно системе проверки орфографии).

Сама по себе альтернативность результатов распознавания очевидна и обусловлена хранением коллекций букв вместе с оценками соответствия. Словарный контроль с использованием словарной базы приводил к изменению этих оценок. В итоге применение словаря позволило реализовать схему дораспознавания символов.

Таким образом, рассматриваемая система обладает свойством самообучения. Она самосовершенствуется в процессе работы, настраиваясь на конкретный текст. Критерий качества распознавания, необходимый при построении алгоритма самообучения, формируется с помощью конкретизирующих параметров, словаря, контекста.

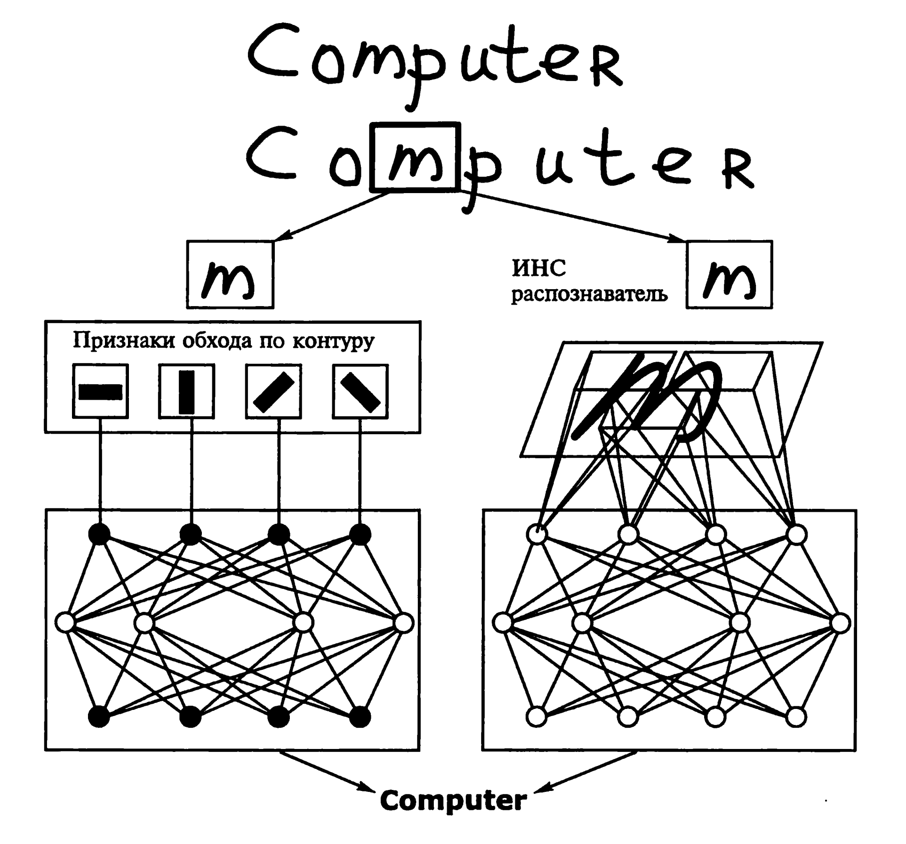


Рис. 4.11. Схема распознавания текста с помощью персептрона с предварительным выделением признаков по методу Паркса (слева) и путем непосредственного сканирования изображения (справа)

Последняя версия CuneiForm 2000 отличается, главным образом, тем, что в ней используется несколько алгоритмов распознавания на основе нейронных сетей. Их применение, по утверждению разработчиков, повысило качество распознавания текстов на 60 %.

Заметим, что на вход нейросети могут подаваться признаки образов, выделенные, например, по методу Паркса, или результаты непосредственного сканирования. Эти две схемы использования нейросети схематично изображены на рис. 4.11.

4.5. ИСПОЛЬЗОВАНИЕ ГЕОМЕТРИЧЕСКИХ ИНТЕРПРЕТАЦИЙ

Задаче распознавания образов по признакам можно придать геометрическую интерпретацию. Например, если для характеристики образа человека использовать всего два признака — рост и вес, то в

системе координат рост—вес множество жителей европейского континента займет область, несколько отличную от множества жителей азиатских стран. Множество азиат *A* расположится несколько левее и выше множества европейцев *B* (рис. 4.12, *а*).

Изображенные на рис. 4.12, *а* множества можно рассматривать как эталонные и использовать для распознавания (классификации) новых объектов (людей), сравнивая расстояния от точки *R*, характеризующей исследуемый объект, до середины эталонных множеств *A* и *B*. Эталонные множества могут пересекаться между собой, а могут находиться на значительном расстоянии друг от друга. Естественно, что пересекающиеся эталонные области менее пригодны для решения задачи распознавания образов.

Ю. В. Девингталем [9, 10] показано, что эталонные множества всегда могут быть отделены друг от друга с помощью преобразований системы координат и, таким образом, между ними может быть проведена некоторая гиперплоскость, разделяющая эти множества (рис. 4.12, *б*). Рассмотрим механизм преобразования системы координат более подробно. Как известно, в *N*-мерном пространстве расстояние между двумя точками *a* и *b* с координатами a_n, b_n ($n = 1, 2, ..., N$) измеряется евклидовой метрикой

$$d(a,b) = \sqrt{\sum_{n=1}^{N}(a_n - b_n)^2}. \quad (4.3)$$

В данную формулу удобно ввести весовые коэффициенты

$$d(a,b) = \sqrt{\sum_{n=1}^{N}w_n^2(a_n - b_n)^2}. \quad (4.4)$$

Весовые множители нужны для отражения того факта, что отдельные признаки объектов имеют разную степень важности для

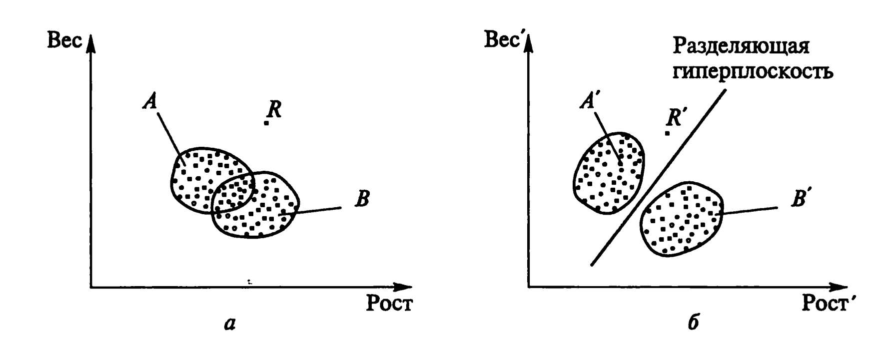


Рис. 4.12. Эталонные множества *A* и *B* до (*а*) и после (*б*) преобразования координат:

R — объект, подлежащий классификации

распознавания образов. Например, при распознавании такой болезни, как шизофрения, температура больного не имеет существенного значения, и для нее можно принять $w_n = 0$.

При геометрической трактовке распознавания образов прибегают к интерпретации процесса в виде некоторого преобразования системы координат, при котором объекты одного класса сжимаются, а множества различных классов удаляются друг от друга. Существуют линейные и нелинейные способы преобразования. В общем виде линейное преобразование пространства признаков задается матрицей

$$\boldsymbol{W} = \begin{Vmatrix} w_{11} & w_{12} & \dots & w_{1N} \\ w_{21} & w_{22} & \dots & w_{2N} \\ . & . & . & . \\ w_{N1} & w_{N2} & \dots & w_{NN} \end{Vmatrix}. \qquad (4.5)$$

Если в исходной системе координат заданы векторы объектов $\boldsymbol{a}$ $(a_1, a_2, \dots, a_N)$ и $\boldsymbol{b}$ $(b_1, b_2, \dots, b_N)$, то преобразованные векторы $\boldsymbol{a}'$ и $\boldsymbol{b}'$ определяются соотношениями:

$$\boldsymbol{a}' = \boldsymbol{aW}; \quad a'_n = \sum_{i=1}^{N} a_i w_{in},$$

$$\boldsymbol{b}' = \boldsymbol{bW}; \quad b'_n = \sum_{i=1}^{N} b_i w_{in}.$$

Евклидово расстояние между элементами в преобразованном пространстве

$$d(\boldsymbol{a}', \boldsymbol{b}') = \sqrt{\sum_{n=1}^{N} (a'_n - b'_n)^2} = \sqrt{\sum_{n=1}^{N} \left[\sum_{i=1}^{N} w_{in} (a_i - b_i) \right]^2}. \qquad (4.6)$$

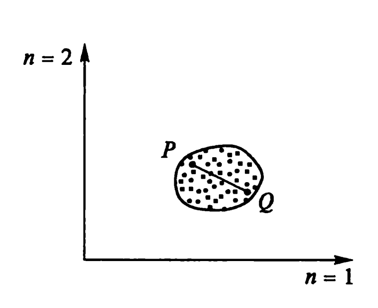


Рис. 4.13. К задаче сжатия множества в n-мерном пространстве

Диагональные элементы матрицы преобразований $\boldsymbol{W}$ определяют масштабные коэффициенты сжатия вдоль координатных осей, а остальные элементы матрицы — поворот координатных осей. В частном случае, если в матрице преобразований все недиагональные элементы равны нулю, т. е. $w_{ij} = 0$, то равенство (4.6) принимает вид

$$d(\boldsymbol{a}', \boldsymbol{b}') = \sqrt{\sum_{n=1}^{N} w_{nn}^2 (a_n - b_n)^2}, \qquad (4.7)$$

что совпадает с (4.4).

Рассмотрим частный случай, когда поворот координатных осей не производится, а изменяется только масштаб. Требуется определить такие коэффициенты сжатия, чтобы в новой системе координат расстояния между объектами множества были минимальными, т.е. чтобы было минимальным расстояние между текущими точками P и Q множества (рис. 4.13):

$$D^2 = \frac{1}{M(M-1)} \sum_{p=1}^{M} \sum_{q=1}^{M} \sum_{n=1}^{N} w_{nn}^2 (x_{np} - x_{nq})^2, \tag{4.8}$$

где p — номер точки P; q — номер точки Q.

Сумма в (4.8) делится на $M(M-1)$, а не на M^2, так как при $p = q$ расстояние между точками равно нулю.

Для решения поставленной задачи требуется наложить дополнительные условия на весовые коэффициенты. Например, можно потребовать, чтобы выполнялось равенство

$$\sum_{n=1}^{N} w_{nn} = 1. \tag{4.9}$$

Это требование наиболее распространено и означает, что весовые множители могут меняться от 0 до 1 и пространство признаков будет сжиматься по осям. В процессе преобразования происходит дискриминация несущественных признаков, у которых весовые коэффициенты значительно меньше единицы.

Помимо (4.9) используют и другие условия, например

$$\prod_{n=1}^{N} w_{nn} = 1. \tag{4.10}$$

В этом случае площади (объемы), занимаемые областями классов признаков до и после преобразования координат, равны друг другу.

Итак, задача обучения состоит в минимизации функционалов (4.8), составленных для множеств A и B (обозначим их D_A^2 и D_B^2) с дополнительным условием (4.9) или (4.10). Эта задача решается путем введения множителя Лагранжа λ, объединяющего функционалы с одним из дополнительных условий:

$$F(w_{nn}) = D_A^2 + D_B^2 - \lambda(\sum_{n=1}^{N} w_{nn} - 1); \tag{4.11}$$

$$F(w_{nn}) = D_A^2 + D_B^2 - \lambda(\prod_{n=1}^{N} w_{nn} - 1). \tag{4.12}$$

Продифференцировав обобщенные функционалы (4.11) или (4.12) по w_{nn} и приравняв производные к нулю, получим систему алгебраических уравнений относительно w_{nn}.

После того как обучение выполнено (весовые множители определены, пространство преобразовано), решается задача собственно распознавания.

Пусть даны два эталонных множества A и B (см. рис. 4.12, *а*) и требуется определить, к какому из них относится некоторый объект R. Для решения этой задачи в преобразованном пространстве вычисляется среднеквадратичное расстояние от R до координат каждого множества:

$$d_A^2 = \frac{1}{M_A} \sum_{i=1}^{M_A} \sum_{n=1}^{N} (x'_{Rn} - x'_{Ain})^2; \qquad (4.13)$$

$$d_B^2 = \frac{1}{M_B} \sum_{i=1}^{M_B} \sum_{n=1}^{N} (x'_{Rn} - x'_{Bin})^2, \qquad (4.14)$$

где M_A — число точек эталонного множества A; M_B — число точек эталонного множества B.

Решающее правило состоит в следующем:

$$R \in A, \text{ если } d_A^2 < d_B^2;$$
$$R \in B, \text{ если } d_A^2 > d_B^2,$$

причем в каждой из функций $d_A{}^2$ и $d_B{}^2$ коэффициенты w_{nn} ищутся применительно к первому или второму множеству эталонов.

Контрольные вопросы

1. Опишите принцип действия пандемониума Селфриджа.
2. Чем различаются между собой демоны понимания в пандемониуме Селфриджа?
3. Каким образом происходит обучение пандемониума Селфрижда?
4. Опишите принцип действия персептрона Розенблатта.
5. Перечислите методы распознавания символов.
6. В чем заключается предварительная обработка изображений?
7. В чем состоит идея цепного кода?
8. Для чего нужны операции поверхностного разрушения и сжатия?
9. Каким образом осуществляется выявление признаков изображения по методу Паркса?
10. Назовите основные принципы работы системы FineReader.
11. Каким образом осуществляется обучение системы CuneiForm?
12. Приведите геометрическую интерпретацию признакового распознавания.
13. С какой целью производятся преобразования координат при распознавании с помощью евклидовых пространств?
14. Дайте математическую формулировку линейного преобразования координат. Какую роль в преобразовании выполняют диагональные и недиагональные коэффициенты матрицы преобразующих коэффициентов?

ГЛАВА 5
ИНТЕЛЛЕКТУАЛЬНЫЕ ИГРЫ

5.1. ПОНЯТИЯ ИГРЫ И ДЕРЕВА ВОЗМОЖНОСТЕЙ

Интеллектуальные игры — это одна из областей искусственного интеллекта, где оптимистические прогнозы ученых 50-х годов прошлого века, хотя и с большим опозданием, но полностью сбылись. В 1998 г. в Нью-Йорке в матче Deep Blue против Гарри Каспарова компьютер впервые победил чемпиона мира по шахматам. Матч состоял из шести партий и завершился со счетом 3,5 на 2,5 в пользу компьютера.

Помимо эмоционально-развлекательного и философского значения интеллектуальные игры представляют еще и практический интерес для развития самой теории искусственного интеллекта. Дело в том, что в современных программах-игроках наиболее полно удалось реализовать центральную идею искусственного интеллекта — обучение, самообучение и самоорганизацию компьютерных программ. Кроме того, понятие «игра» имеет более широкое значение. Игрой можно считать многие экономические, политические, военные и другие конфликты.

Проблемой создания игровых программ, в частности, шахматных, занимались многие ученые-кибернетики, такие как Тьюринг, Стречи, Шеннон, Нильсон. Принципы работы, предложенные каждым из разработчиков, опираются на исследования дерева возможных продолжений игры. Корневая вершина дерева возможностей представляет собой текущее положение фигур на шахматной доске, а работа программы состоит в выборе очередного хода.

В середине партии у игрока обычно имеется около 30 возможных вариантов следующего хода. Возникающие в результате их перебора конфигурации представляются как дочерние вершины для данной корневой вершины. В каждой из дочерних вершин возможно около 30 ответов противника, так что для изображения результирующих конфигураций потребуется еще около 900 вершин и т.д. Дерево быстро разрастается (рис. 5.1), что приводит к комбинаторному взрыву.

Все вершины могут быть двух типов. В одних очередной ход предстоит делать компьютеру, в других — его противнику. Первые называют *альфа-вершинами*, вторые — *бета-вершинами*. Таким об-

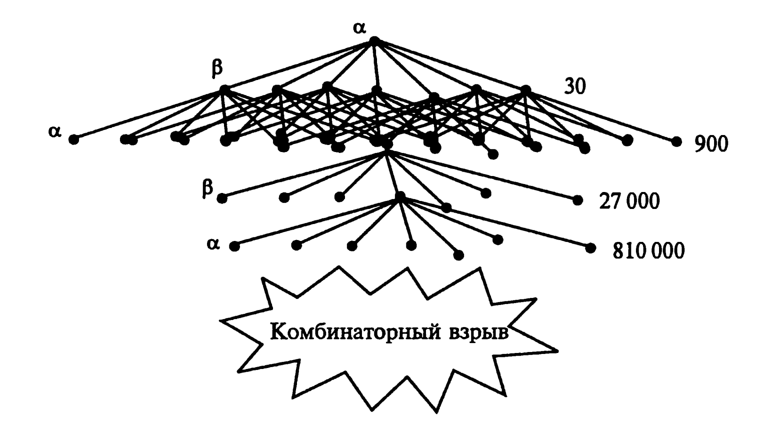


Рис. 5.1. Дерево возможных продолжений шахматной игры

разом, дерево возможностей представляет собой чередующиеся слои альфа- и бета-вершин.

Если бы дерево можно было обследовать полностью, т. е. вплоть до листьев, представляющих собой все возможные окончания в данной игре, то имелась бы возможность выбрать ход, обеспечивающий для компьютера выигрыш независимо от реакций противника. Такая возможность имеется в простейших играх, таких как крестики—нолики, каллах и др. В интеллектуальных играх типа шахмат удается построить и просмотреть лишь небольшую часть дерева возможностей. В этом случае говорят, что дерево возможностей подвергается подрезке, а конечные вершины, ниже которых дерево отсечено, называют *терминальными* вершинами.

5.2. МЕТОДЫ ПОДРЕЗКИ ДЕРЕВА ВОЗМОЖНОСТЕЙ

В программах-шахматистах для каждой вершины обычно определяются числовые оценки силы позиций каждого из партнеров. Такую оценку называют оценивающим полиномом, или *оценивающей функцией*:

$$S = k_1a_1 + k_2a_2 + k_3a_3 + \ldots, \qquad (5.1)$$

где k_1, k_2, k_3 — весовые коэффициенты; a_1, a_2, a_3 — некоторые признаки силы позиции.

Обычно оценивающая функция равна нулю, когда позиции партнеров равноценны; положительна, когда преимущество за компьютером, и отрицательна, когда преимущество за противником.

Важной компонентой любой оценивающей функции является материальное соотношение или перевес фигур (в формуле (5.1)

это a_1). При этом каждой фигуре придается определенное значение ценности.

Другой важной компонентой оценивающей функции (a_2) должна быть некая мера подвижности фигур, развитости позиции. Критерием подвижности может быть число допустимых ходов, имеющихся у игрока. Далее идет оценка контроля центра шахматной доски (a_3) и т.д.

После того как произведена оценка каждой терминальной вершины (конечной вершины при заданной глубине обследования дерева возможностей), выполняется перенос результатов этих оценок вверх по дереву (в направлении корня дерева). Метод, которым это достигается, называется *минимаксным переходом*. Он заключается в следующем. Для альфа-вершин принимается значение, равное наибольшей из найденных оценок для дочерних вершин. Такое решение абсолютно оправданно, поскольку, опираясь на такие оценки, компьютер сделает правильный для себя ход. Наоборот, для бета-вершин принимается наименьшая из оценок для дочерних вершин, поскольку можно предполагать, что противник сделает ход, наименее выгодный для компьютера. В итоге некоторое оценочное значение будет приписано и корневой вершине. Поскольку она является альфа-вершиной, это значение будет наибольшим среди значений для дочерних вершин. Ход, который выбирает компьютер, преобразует существующую на шахматной доске конфигурацию, представленную корневой вершиной, в конфигурацию, представленную той дочерней вершиной, из которой было взято значение оценки для корневой вершины.

Аналогичный подход может быть применен в программировании автоматов для других игр. Нильсоном была предложена оценивающая функция для игры в крестики—нолики. Пусть компьютер ставит крестик, а его противник — нолик. Если конфигурация, которую предстоит оценить, содержит три крестика подряд (т.е. компьютер выигрывает), то оценивающая функция принимает наибольшее значение, например +10. Если подряд стоят три нолика, то оценивающая функция принимает наименьшее значение, например –10. Для ситуаций, которые не являются выигрышными ни для одного из игроков, оценивающую функцию Нильсон предложил вычислять так: число строк, столбцов и диагоналей из восьми возможных, все еще открытых для компьютера, т.е. не заблокированных ноликами, минус число строк, столбцов и диагоналей, все еще открытых для противника, т.е. не заблокированных крестиками. Примеры оценивающих функций Нильсона для различных ситуаций приведены на рис. 5.2.

Как видно из рис. 5.3, использование такой оценивающей функции не годится, если дерево имеет единичную глубину, т.е. состоит только из корневой вершины и ее дочерних вершин. Для каждого из пяти возможных ходов оценивающая функция оказы-

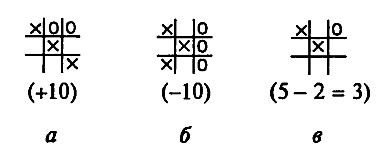


Рис. 5.2. Примеры оценивающих функций Нильсона: *а* — выиграл компьютер; *б* — выиграл противник; *в* — преимущество за компьютером

вается равной нулю. Таким образом, предлагаемый метод не позволяет выбрать единственно правильный ход (*г*) — в левый нижний угол.

Однако, как видно из рис. 5.4, просмотр вперед на два уровня позволяет компьютеру выбрать хороший ход. Под каждой терминальной вершиной на рис. 5.3 и 5.4 указано в скобках значение оценивающей функции, вычисленной по методике Нильсона. Для нетерминальных вершин аналогичные значения получаются путем применения минимаксной процедуры.

Деревья, представленные на рис. 5.3 и 5.4, подрезаны путем отсечения их на определенной глубине. Однако установлено, что можно добиться более высокого качества игры, если использовать другие методы отсечения, получившие общее название *методов прямого усечения*.

Например, в одном из методов прямого усечения выделяют *мертвые*, или *спокойные*, позиции, тогда как другие позиции определяют как *живые*, или *беспокойные*. Вершина с большей вероятностью может быть принята за терминальную (конечную), если представляемая ею конфигурация классифицируется как мертвая. Разделение же позиций на живые и мертвые осуществляется с помощью эвристических правил. Например, позиция классифицируется как живая, если существует угроза взятия фигур (речь идет о шашках или шахматах).

При использовании прямого усечения дерева нельзя быть абсолютно уверенным, что результат поиска окажется таким же, как если бы дерево не подвергалось подрезке. Метод обратного усечения более надежен. Его иногда называют процедурой альфа—бета. Как мы видели, поиск по дереву включает в себя два этапа: построение дерева возможностей с последующим припи-

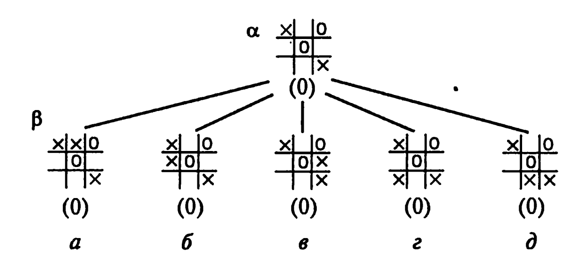


Рис. 5.3. При единичной глубине дерева компьютер не видит единственно правильного хода — *г*; *а*, *б*, *в*, *д* — неправильные ходы компьютера

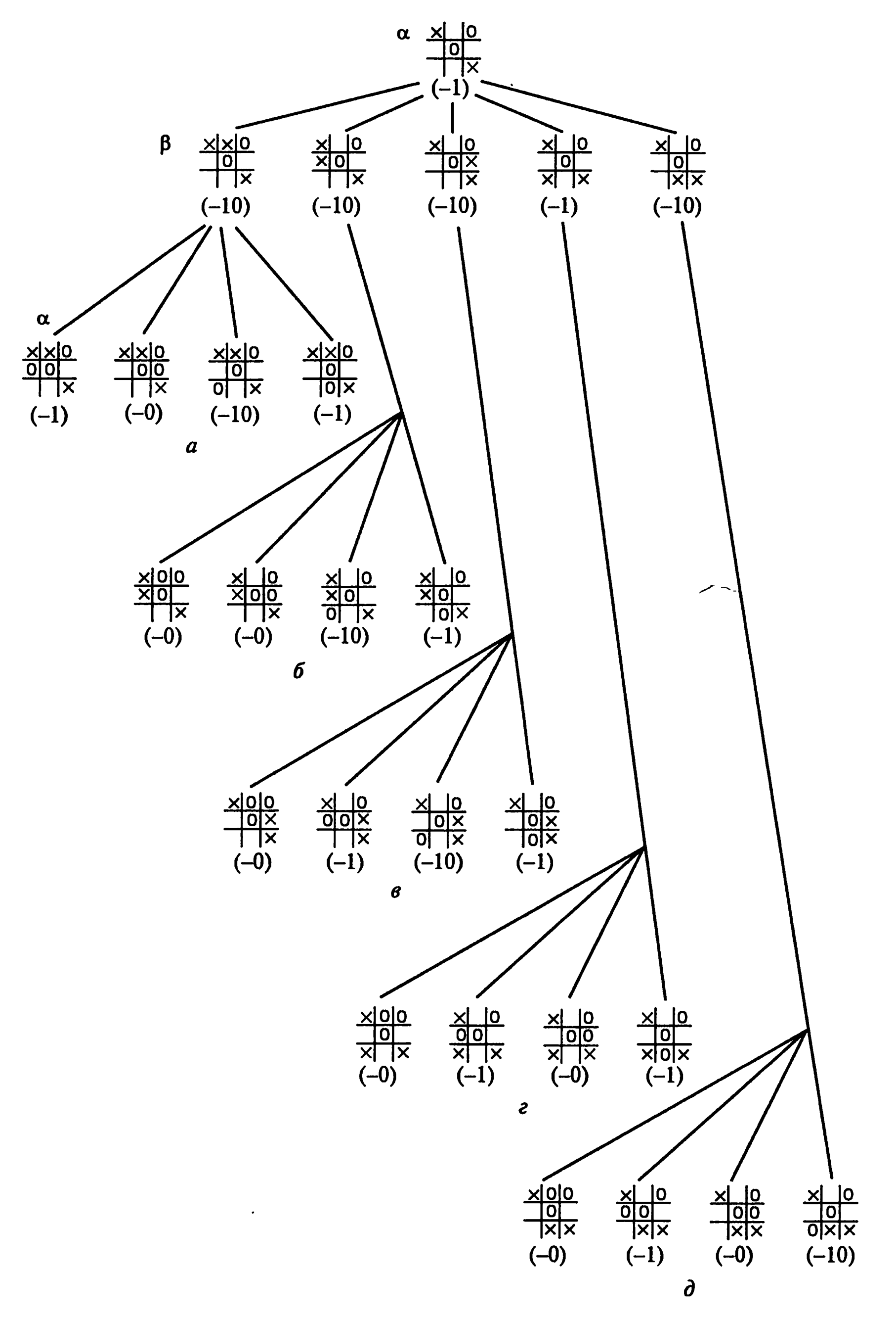


Рис. 5.4. Выбор хода в игре в крестики—нолики при глубине дерева, равной двум (все варианты, кроме варианта *г*, означают поражение компьютера)

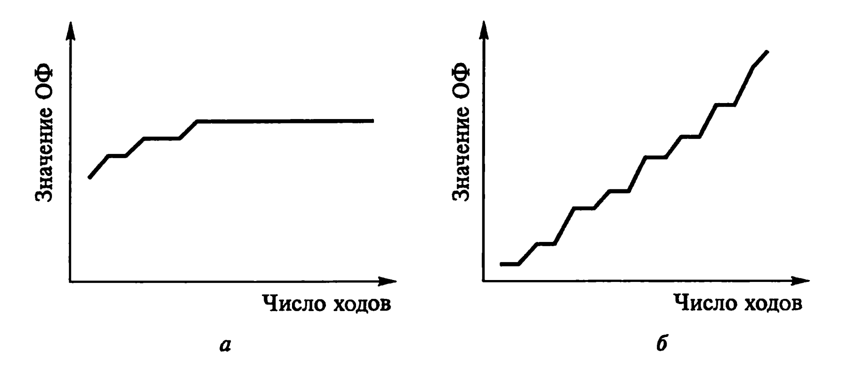


Рис. 5.5. Типичное поведение оценивающей функции (ОФ) в «мертвой» (*а*), «живой» (*б*) альфа-вершине

сыванием его терминальным вершинам числовых значений оценочной функции, а затем применение минимаксной процедуры для передачи значений вверх по дереву. Процедура альфа—бета предполагает объединение этих двух этапов так, чтобы значения оценочных функций связывались с вершинами по мере формирования дерева. Применение минимаксной процедуры приводит к тому, что оценивающие функции для каждой альфа-вершины с ростом дерева могут только увеличиваться, а для каждой бета-вершины — только уменьшаться. Наблюдение за динамикой изменения оценивающих функций (рис. 5.5) дает возможность понять, что некоторые из еще не построенных вершин никак не могут повлиять на конечный результат, и отказаться от целых ветвей. Таким образом, обратное усечение состоит в отказе от построения неперспективных вершин, причем распознавание таких вершин ведется путем изучения динамики их оценивающих функций.

В настоящее время эффективность шахматных программ постоянно возрастает. Существуют программы, которые легко обыгрывают шахматистов-дилетантов и бросают вызов профессионалам. Тем не менее, шахматные программы работают по принципу «грубой силы», который основан на построении и исследовании дерева возможностей.

Очевидно, что подобный способ выбора хода не соответствует тому, как поступает в этом случае человек. Шахматисты обычно предпринимают весьма ограниченный поиск, учитывая лишь небольшое число возможностей. Обсуждая между собой шахматные партии, шахматисты говорят не о поиске по дереву возможностей, а прибегают к таким эмоциональным понятиям, как атака, защита, угроза, нападение. Часто шахматисты вообще не рассматривают варианты, а делают только один единственно правильный ход.

Поэтому наряду с увеличением мощности шахматных программ появилось новое направление, связанное с насыщением их эври-

стиками, заимствованными из шахматных партий, т. е. оснащение шахматных программ базами знаний. Немаловажную роль играет применение методов обучения и самообучения игровых программ.

5.3. Идеи обучения игровых программ

Представляет интерес программа для игры в шашки, разработанная Артуром Сэмюэлем (см. [48]). В этой программе Сэмюэлю удалось реализовать две формы обучения: накопление и обобщение.

Накопление сводится к хранению в памяти компьютера большого числа конфигураций на шашечной доске из тех, что реально (а не гипотетически) возникают в ходе шашечных игр. Вместе с каждой конфигурацией в памяти хранится также ее числовая оценка, которая получилась путем построения дерева, применения оценивающей функции к терминальным вершинам и передачи значений вверх по дереву посредством минимаксной процедуры. Имея в памяти некоторое множество конфигураций вместе с их оценками, программа в процессе работы ищет соответствие между конфигурацией, отвечающей каждой из вершин дерева, и конфигурациями из числа запомненных. Если такое соответствие установлено, то хранимая в памяти оценка передается в эту вершину. В результате отпадает необходимость строить какую-либо ветвь, которая могла бы возникнуть под этой вершиной.

Таким образом, накопление позволяет либо экономить время, либо достичь лучшего качества игры за то же время путем использования несколько большего дерева.

Естественно, размер списка конфигураций, который может храниться в памяти и использоваться, ограничен сверху. А. Сэмюэль построил свою программу так, что наименее употребляемые конфигурации вычеркиваются, а часто встречающиеся остаются в памяти компьютера.

Другая форма обучения, использованная А. Сэмюэлем, — *обобщение*. Оно позволяет программе в ходе игры улучшать свои оценивающие функции. Обычно оценивающая функция представляет собой полином; в простейшем виде это полином первой степени, или взвешенная сумма

$$S = k_1 a_1 + k_2 a_2 + k_3 a_3 + \dots,$$

где a_1, a_2, a_3 — величины различных вычисляемых критериев, таких как материальное соотношение, подвижность, контроль центра шахматной доски и пр.; они взвешиваются по отношению друг к другу с помощью коэффициентов k_1, k_2, k_3.

Полином может быть также и более высокой степени относительно переменных a, например,

$$S = k_1 a_1 + k_2 a_2 + k_{11} a_1^2 + k_{12} a_1 a_2 + \dots \qquad (5.2)$$

Качество игры зависит от подходящего выбора весовых коэффициентов k_1, k_2, k_3, ..., и обобщение является средством их подгонки, обеспечивающей улучшение игры. Метод обобщения представляет собой пример оптимизации с использованием процедуры, называемой «подъем в гору». Имеется начальный набор значений k_1, k_2, k_3, ..., и в каждый момент времени эти коэффициенты определяют рабочую точку. Рабочая точка перемещается в пределах многомерного пространства (рис. 5.6) по мере подгонки величин весовых коэффициентов в поисках положения, в котором оптимизируется определенная реакция или целевая функция.

Чтобы воспользоваться методом подъема на гору, следует дать программе возможность сыграть некоторое число игр с определенным партнером, выбрав какое-то начальное множество коэффициентов k_i, а затем сыграть еще некоторое число игр, сделав пробные изменения в положении рабочей точки. Если программа во втором множестве игр выигрывает чаще, то принимается новое значение k_i. В противном случае происходит возвращение к старой величине и производится какое-то новое пробное изменение.

Очевидно, указанный путь поиска k_i весьма далек от совершенства. Во-первых, этот путь предполагает очень медленное движение. Во-вторых, поскольку партнер не может играть абсолютно ровно, необходимо, чтобы два указанных множества игр были достаточно емкими.

Поэтому А. Сэмюэлем был предложен другой путь нахождения весовых коэффициентов во время игры, который основан на том, что качество игры растет с увеличением глубины просмотра дерева возможностей. Если может быть найдено средство вычисления оценочной функции, обеспечивающее точное совпадение переданного назад по дереву (с большой глубиной) значения оценочной функции с результатом его прямого (с небольшой глубиной) определения, то такая оценка должна быть равнозначна изучению всего полностью построенного дерева игры.

Если S — результат прямой оценки с помощью оценочной функции, а S_b — результат передачи оценки по дереву (с большой глубиной), то можно считать их разность ошибкой e, где

$$e = S - S_b. \quad (5.3)$$

Сэмюэль сделал так, что в его программе вычислялась корреляция между e и a_1, a_2 и т. д. Положительная корреляция между e и любым значением a_i указывает, что соответствующий коэффициент k_i следует уменьшить, а отрицательная корреляция означает, что его надо увеличить.

При применении указанного метода требуется уделить внимание обеспечению его устойчивости. Для повышения устойчивости Сэмюэль фиксировал один из весовых коэффициентов, тогда как

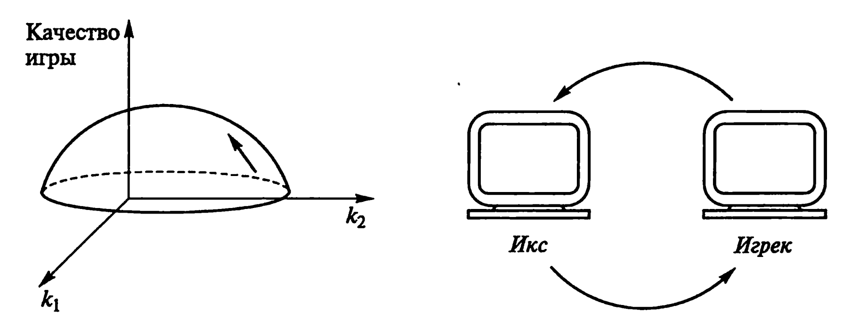


Рис. 5.6. Качество игры как многомерная функция весовых коэффициентов

Рис. 5.7. Компьютеры обучают друг друга

другие коэффициенты изменялись. Обычно это был наиболее важный параметр, оценивающий материальное соотношение, поскольку разумно полагать, что игроку всегда выгодно, чтобы его фигуры на доске сохранялись.

Таким образом, А. Сэмюэлем был создан алгоритм программы, обладающий свойством самообучения (обучение без учителя). Насколько нам известно, это и была первая в мире действующая самообучающаяся программа.

Однако А. Сэмюэль пошел еще дальше. Он держал в своей программе больший ассортимент критериев (a_1, a_2 и т.д.), чем тот, что допускался для использования в конкретной оценивающей функции. Используемое множество критериев видоизменялось во время игры: если какое-то из значений весовых множителей k_i оставалось близким к нулю в течение длительного времени, то тот компонент оценивающей функции, к которому относился этот коэффициент, изымался из рабочего множества, а на его место ставился другой из числа ожидавших своей очереди. Изъятый критерий добавлялся к множеству ожидавших своей очереди и мог быть впоследствии заново внесен в оценивающую функцию.

Возможность изменения множества критериев a_i придает данному методу обучения новый характер. Теперь его можно воспринимать как некую самоорганизующуюся систему, способную изменять в процессе обучения не только свои параметры, но и структуру.

Следующая идея А. Сэмюэля была воистину гениальной. Он замкнул игровую программу саму на себя — организовал работу программы таким образом, что она могла вести игру и самообучаться непрерывно днем и ночью, причем имитировала одновременно двух игроков — *икс* и *игрек* (рис. 5.7) Игроку *икс* разрешалось модифицировать свою оценивающую функцию путем обобщения, тогда как игрок *игрек* пользовался фиксированной оценивающей функцией. Когда *икс* выигрывал игру, игрок *игрек* копировал оце-

нивающую функцию у игрока *икс*. Если же игрок *игрек* выигрывал подряд три игры, то его оценивающая функция копировалась игроком *икс*. Это гарантировало возможность возвращения игрока *икс* к прежнему положению в том случае, если процесс подгонки параметров происходил в нежелательном направлении.

Таким образом, А. Сэмюель создал программу, которая позволяла не только правильно играть в шашки, но и улучшать стратегию игры, используя опыт, накопленный в предыдущих партиях. Вначале А. Сэмюель с легкостью обыгрывал машину, но машина (IBM 704) начала быстро совершенствоваться. Вскоре она достигла такого уровня, что выигрывала у А. Сэмюеля все партии подряд.

По существу этот факт, произошедший в 1959 г., открыл новую эру в истории человечества — эру самонастраивающихся, самообучающихся, самоорганизующихся компьютеров. Компьютер сам, без помощи человека, научился совершенствоваться. Нетрудно догадаться, к чему это может привести в недалеком будущем, и не в ад ли ведет дорога, вымощенная благими намерениями?

Оставив затронутую тему философам и фантастам, укажем на недостаток современных обучающихся игровых программ.

Дело в том, что в современных игровых программах, как правило, реализованы сразу две парадигмы обучения — с учителем и без него. Понятно, что результат обучения таких программ зависит от конкретного учителя. И очень часто вместо того чтобы учиться играть в игру, такие программы учатся обыгрывать учителя. Яркий пример тому мы видели несколько лет назад: после победы DeepBlue над Гарри Каспаровым программисты IBM отказались от матча с другими гроссмейстерами. В результате чемпион мира заявил, что программа просто была «натаскана» на его партиях, она изучила его стиль и потому просто не способна конкурировать с другими гроссмейстерами.

Современная шахматная программа заведомо может научиться выигрывать у любого, но после этого ей придется некоторое время перестраиваться под другого соперника. Но, если вдуматься, такой же недостаток имеет любой начинающий шахматист — он учится и в процессе обучения перенимает тактику и стратегию игры своего учителя.

Остается открытым вопрос о способности игровых программ к творчеству. Смогут ли они вырабатывать принципиально новые решения, стратегии, стили, манеры, или это навсегда останется прерогативой человека?

Но компьютерное творчество — это тема следующей главы.

Контрольные вопросы и задания

1. Что представляет собой дерево возможностей?
2. Какие вершины дерева возможностей называются терминальными?

3. По каким принципам осуществляется подрезка деревьев возможностей?

4. По каким принципам формируется оценивающая функция?

5. Объясните суть минимаксного перехода.

6. Назовите способы улучшения оценивающих функций.

7. В чем состоят идеи А. Сэмюэля, касающиеся самообучения и самоорганизации игровых программ?

8. Каким образом происходит обучение современных шахматных программ?

9. Почему победа компьютера над чемпионом мира по шахматам 1998 г. была поставлена под сомнение? Согласны ли вы с этим?

ГЛАВА 6
КОМПЬЮТЕРНОЕ ТВОРЧЕСТВО

6.1. ФИЛОСОФСКИЕ АСПЕКТЫ ТВОРЧЕСТВА

Любое произведение искусства может быть закодировано в виде конечного числа цифр. Например, каждое слово поэмы состоит из букв, которые могут быть закодированы 33 цифрами. Ясно, что при таком соответствии одна длинная строка цифр может рассматриваться как кодированная запись поэмы.

Аналогично обстоит дело в живописи. Полотно картины можно расчертить на мельчайшие клетки и цвет каждой клетки закодировать цифрами. Такое представление произведений живописи, в отличие от оригиналов, не подвержено разрушительному действию времени и может храниться веками.

То же самое в музыке. Из анализа Фурье известно, что все звучание музыкального произведения, от первой ноты до последней, может быть представлено одной единственной кривой на экране осциллографа. Кривую можно с любой степенью точности кодировать цифрами.

Таким образом, любое произведение искусства в любой области можно представить в виде набора конечного числа цифр. Число возможных комбинаций этих цифр огромно, но не бесконечно. Поэтому можно вообразить себе библиотеку, содержащую все возможные комбинации цифр. Подавляющее большинство комбинаций цифр в переводе на буквы, цвета и звуки не имеют никакого смысла. Но среди этих комбинаций есть и такие, которые имеют смысл и являются тем, что мы называем произведениями искусства. Существуют ли алгоритмы, которые позволят компьютеру выбрать из множества бессмысленных вариаций те, которые являются гениальными поэмами, картинами, симфониями?

Первые попытки создания таких алгоритмов относятся к XVII в. Известен, например, пятисотстраничный трактат немецкого иезуита Афанасиуса Кирхера «Универсальная музургия, или великое искусство созвучий и диссонансов». А. Кирхер был учеником Луллия и рассматривал музыкальную композицию как комбинаторную задачу. Его идеи были реализованы в виде устройства, напоминающего механическую экспертную систему Луллия (см. подразд. 1.1). Ныне это устройство хранится в Кембриджском музее.

В начале XVIII в. вопросами механического сочинения музыкальных произведений с помощью таблиц и игральных костей занимались многие известные композиторы, такие как Бах, Гайдн, Моцарт.

Рассмотрим вопрос создания произведений искусств с использованием известных нам современных методов искусственного интеллекта.

Мы уже сталкивались с методами математического моделирования в различных естественных науках, таких как физика, метеорология, экономика, механика сплошных сред, электроника и пр. Роль математического моделирования в жизни современной цивилизации переоценить трудно, причем круг проблем, решаемых этим методом, постоянно растет.

Напомним, что модель — это «черный ящик», в который вводятся входные и выводятся выходные параметры. Модель является намеренно упрощенной схемой некоторого реального объекта, системы, процесса. Но на основе исследования модели получают рекомендации для решения реальных проблем.

Математическая модель может существовать в виде логических программ, переводимых на язык ЭВМ. Математическую модель, введенную в компьютер, называют компьютерной моделью.

Существуют общие принципы построения моделей. Вот некоторые из них.

Для построения модели необходимо:

а) выявить релевантные (существенные) факторы, т.е. факторы, которые могут сказываться на результатах решения данной проблемы или на исходе рассматриваемого процесса;

б) выбрать факторы, которые могут быть описаны количественно;

в) объединить факторы по общим признакам и сократить их перечень, выявить инварианты (о них речь пойдет дальше);

г) установить количественные соотношения между выбранными факторами и инвариантами.

Факторы, которые по своей природе не могут быть выражены количественно, так же, как и несущественные факторы, исключаются из рассмотрения.

При математическом моделировании очень важным этапом является установление инвариантов системы. Поэтому рассмотрим этот вопрос подробнее.

Идея инвариантности состоит в том, что, хотя система в целом претерпевает последовательные изменения, некоторые ее свойства сохраняются неизменными. Существование инварианта в любом множестве неизбежно влечет за собой ограничение разнообразия.

По Эшби, слово «разнообразие» в применении к множеству различающихся элементов употребляется в двух смыслах — как число

различных элементов и как логарифм этого числа по основанию 2. Так, множество *а*, *б*, *б*, *а*, *б*, *б*, *б*, *а* содержит восемь элементов, имеет разнообразие в два элемента в первом смысле и в $\log_2 2 = 1$ бит во втором смысле. Разнообразие 52 игральных карт равно $\log_2 52 = 5{,}7$ бит.

Существование инварианта во множестве явлений говорит об ограничении разнообразия. Поэтому теория инвариантов — это теория ограничения разнообразия.

Поскольку любой закон природы подразумевает существование некоторого инварианта, то, следовательно, всякий закон природы есть ограничение разнообразия, а так как цель науки есть поиск законов, то наука занимается поиском ограничений разнообразия.

В математике инвариантом называют функцию от преобразуемой величины, не изменяющую своего значения при преобразовании этой величины. Так, расстояние между двумя точками

$$S = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

является инвариантным относительно переноса начала координат и поворота осей на любой угол.

В гидродинамике инвариантами являются критерии подобия — числа Грассгофа, Прандтля, Релея и т.д.

В лингвистике выделяют инварианты стихотворных форм. Например, старинная итальянская форма стиха — сонет — всегда имеет 14 строк. Первая часть его состоит из двух катренов (четверостиший), вторая — из двух терцетов (трехстиший). Стихотворный размер сонета — пятистопный (реже шестистопный) ямб. Форма рифмовки для катренов — две разнозвучные рифмы, а для терцетов — две рифмы, отличающиеся от рифм катренов. Третья строка первого терцета рифмуется со второй строкой второго терцета и т.д. В любом сонете легко обнаружить сохранение указанных инвариантов.

Аналогичным образом инварианты могут быть обнаружены и в других произведениях искусства. Так, в мажорных музыкальных произведениях, характерных для старинных русских мелодий, помимо основного мажорного аккорда всегда присутствуют еще два. Например, в до-мажоре — это обязательно фа-мажор и соль-мажор. А в минорном произведении, характерном для поп-музыки, например в ре-миноре, всегда присутствуют соль-минор и ля-мажор, а также фа-мажор и до-мажор.

Возможность создания тех или иных произведений искусства может быть определена в первом приближении как сознательная или бессознательная способность находить нужные инварианты и комбинировать их для получения желаемого эффекта. Эта способность проявляется художником-творцом в результате обобщения закономерностей всего предшествующего художественного наследия.

Таким образом, мы видим, что искусство, в конечном итоге, преследует ту же цель, что и наука — *выявление инвариантов, установление связи между ними, ограничение разнообразия.*

О возможности моделирования творческой деятельности человека непрерывно идут дискуссии, существуют различные точки зрения, как положительные, так и отрицательные. Попытаемся рассмотреть этот вопрос с математической точки зрения. Что есть творчество с точки зрения математика?

Воспользуемся известной теоремой Геделя. Смысл ее состоит в том, что всякая достаточно мощная формальная непротиворечивая логико-математическая система обязательно содержит формулу, которую в данной системе нельзя ни доказать, ни опровергнуть.

Иначе говоря, если имеем систему аксиом $A_1, A_2, \ldots, A_n$, то в этой системе можно сформулировать предложение B_0, которое невозможно ни доказать, ни опровергнуть при помощи данной системы аксиом. Однако может оказаться, что при добавлении к имеющейся системе аксиом некоторой аксиомы A_{n+1} станет возможным доказать или опровергнуть предложение B_0. Но и в этом случае обязательно найдется еще хотя бы одно предложение B_1, которое невозможно ни доказать, ни опровергнуть при помощи уже расширенной системы аксиом. Систему нужно снова расширять и т.д. Так, геометрия Лобачевского содержит в себе геометрию Евклида, а из теории относительности Эйнштейна, как частный случай, следует ньютоновская механика.

Творчество — это процесс расширения системы, в результате чего невыводимые утверждения становятся выводимыми. Иначе говоря, если некоторая задача не может быть решена в данной логической системе, то необходимо искать другую систему, логически более мощную. Тогда творчество заключается в расширении системы, увеличении ее логической мощи, ее логического «богатства», что дает возможность решения новых задач, не решаемых в старой системе.

Итак, можно дать два определения творчества.

1. Это поиск инвариантов и соотношений между ними.

2. Это расширение логической системы с целью решения новых задач.

Так с математической точки зрения можно представить процесс творчества.

Не меньшее методологическое значение для понимания и моделирования процесса творчества имеют теоремы Мак-Каллока и Питтса — основателей направления, называемого нейрокибернетикой. Этими авторами введено понятие математического нейрона. Если нейрон является основной рабочей клеткой коры больших полушарий мозга человека, то математический нейрон есть абстрактный логический элемент, в котором формально отраже-

ны лишь те свойства живого нейрона, которые связаны с переработкой информации. Принцип действия математического нейрона и его возможности для решения практических задач изложены в гл. 3. К теме этой главы имеют отношение теоремы Мак-Каллока — Питтса, смысл которых сводится к тому, что любое функционирование живой нервной ткани, которое можно описать с помощью конечного числа слов в терминах логического исчисления высказываний, может быть описано при помощи искусственной нейронной сети. Таким образом, существует принципиальная возможность создания сети из математических нейронов, способной к творческой деятельности.

Теоремы Мак-Каллока — Питтса представляют собой теоремы существования. Они не говорят о том, как нужно создавать сеть из математических нейронов, чтобы воспроизвести творческую деятельность человека, а только утверждают, что такую сеть принципиально можно построить. В этом состоит методологическое значение теорем Мак-Каллока — Питтса.

Практических же успехов в области моделирования творческой деятельности удалось добиться, следуя другим альтернативным направлением искусственного интеллекта, называемым кибернетикой «черного ящика».

6.2. Моделирование в музыке

Человеческий мозг — это своего рода банк данных и знаний, в котором хранится огромная информация, собранная за всю прожитую человеком жизнь. Доказано, что человеческий мозг никогда и ничего не забывает. Каждый прожитый им день до мельчайших подробностей, как на видеопленку, записывается в память. И эта информация может быть определенным образом извлечена. Кроме того, имеются попытки доказать, что может быть извлечена информация, переданная человеку от предыдущих поколений и из прожитых им ранее жизней.

В памяти композитора существует множество различных мелодий, накопленных им в течение жизни, может быть, переданных с генами от далеких предков. И естественно полагать, что фрагменты этих мелодий, отдельные музыкальные фразы, музыкальные инварианты осознанно или неосознанно используются композитором в его творческом процессе.

Поэтому первое, что нужно сделать при создании модели музыкального творчества (рис. 6.1), это занести в память компьютера как можно больше музыкальных произведений (создать базу данных). Далее, как и в любой интеллектуальной системе, нужно создать базу знаний, состоящую из законов музыкальной гармонии — соотношений между музыкальными инвариантами (соль-

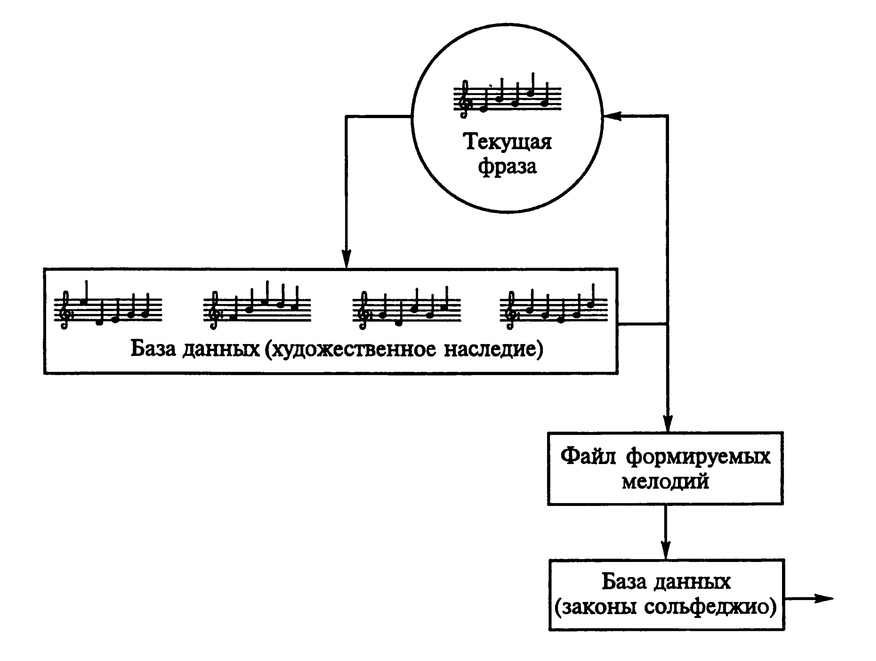


Рис. 6.1. Примерная схема музыкального творчества

феджио). Например, если вы собираетесь написать музыкальное произведение в ре-миноре, то вам необходимо внести основные для этой тональности аккорды — соль-минор, до-мажор, фа-мажор, ля-мажор, и задать приемлемые варианты перебора последовательности этих аккордов.

На вход компьютера надо подать начальное приближение (творческое вдохновение) — музыкальную фразу, состоящую, например, из четырех нот, и заставить компьютер отыскать такое же сочетание нот в одном из хранящихся в его памяти музыкальных произведений. Далее следует прочитать следующую за найденной фразой ноту, приписать ее к текущей музыкальной фразе, а первую ноту из этой фразы выдать в файл формируемых мелодий и вычеркнуть из текущей фразы так, чтобы в ней по-прежнему оставалось четыре ноты. Процесс поиска надо продолжить, анализируя следующие за найденной в памяти компьютера мелодии.

В результате в файле формируемых мелодий сформируется последовательность нот новой мелодии, которая по своему звучанию будет напоминать заложенные в память компьютера известные мелодии, но отличаться от них. Например, если в память закладывались вальсы, то на выходе будет вальс, если марши, то на выходе — марш и т. п.

Алгоритм выбора продолжений мелодий из базы данных можно снабдить эвристическими правилами, регулируя их силу с по-

мощью все тех же коэффициентов доверия. Естественно, что для разных музыкальных стилей будут и разные коэффициенты доверия. Эти коэффициенты можно изменять в процессе обучения, добиваться улучшения качества сочиняемых мелодий.

Изложенная методика компьютерного синтезирования музыки в упрощенной форме отражает процесс творчества композитора и не претендует на полную модель его деятельности. В ней есть база данных, содержащая художественное наследие, база знаний, состоящая из законов музыкальной гармонии (сольфеджио), а также элемент вдохновения в виде начального приближения, которое можно задавать генератором случайных чисел. Есть также возможность совершенствования таланта компьютерного композитора за счет его обучения, например путем модификации коэффициентов доверия используемых правил формирования продолжений мелодии.

Более сложные методики, отражающие также другие стороны музыкального творчества, были предложены в 1955 г. исследователями Иллинойского университета Хиллером и Исааксом. Они провели серию экспериментов, в которых последовательно закладывались законы сольфеджио в базу знаний (гармонизация) и вводились разнообразные музыкальные ритмы и темпы. Был также использован датчик случайных чисел.

На рис. 6.2 изображена блок-схема программы, реализованной на машине «Иллиак». На входе программы генератором случайных чисел задавались целые числа, при помощи которых закодированы нотные знаки. Каждое из чисел пропускалось через последовательность из четырех контрольных схем (*I*—*IV*). Эти схемы пропускали в запоминающее устройство только те числа, которые образовывали правильную (подчиняющуюся заложенным в контрольной схеме ограничениям) мелодическую линию. Законченный период запоминался, а затем выдавался на печать и расшифровывался в виде нот. Если же хотя бы одна из схем задерживала хотя бы одну ноту, то управление вновь передавалось генератору случайных чисел и поиски правильной ноты продолжались. После 50 неудачных попыток подобрать нужную ноту мелодическая линия разрушалась и начинала выстраиваться новая линия.

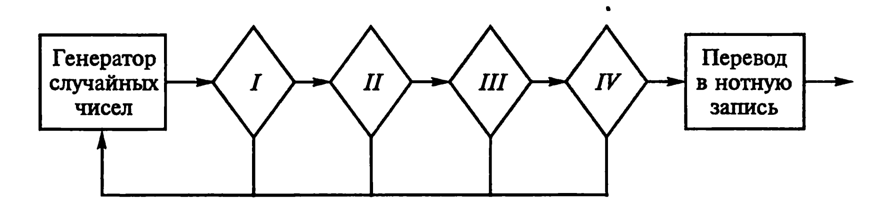


Рис. 6.2. Блок-схема программы компьютерного сочинения музыки

За 1 ч работы машина «Иллиак» создавала 100 мелодий. Так была написана знаменитая «Иллиак-сюита» для струнного оркестра.

Широкую известность в свое время получили музыкальные произведения, сочиненные ЭВМ «Урал-2» по алгоритмам, разработанным советским математиком Р. Х. Зариповым. В его программах также использовался генератор случайных чисел, который выдавал не только код ноты, но и длительность и интервал между нотами. Эти данные проходили контроль на соответствие закономерностям базы знаний — законам музыкальной гармонии, полученным при анализе широкого круга музыкальных произведений.

Программы Р. Х. Зарипова моделировали сочинение мелодий в мажоре или миноре, аккомпанемента к ним, сочинение мелодий на заданный стихотворный текст и ритм.

Принципы, разработанные первыми музыковедами-программистами, в настоящее время закладываются в схемы современных оркестровых электромузыкальных инструментов и широко используются композиторами и музыкантами. Однако этот инструментарий является вспомогательным, так как его применение ограничено сочинением гармонии, аранжировок, сопровождений. Сами же темы сочиняет по-прежнему человек. Дело в том, что создать хорошую простую мелодию неизмеримо сложнее, чем оркестровое произведение в авангардистской манере, перегруженное случайными звукосочетаниями и диссонансами. Когда композитор сочиняет мелодию, которая становится популярной, происходит колоссальный прорыв вперед, так как это открытие нового, не известного ранее соотношения между музыкальными инвариантами.

Современные компьютерные программы могут сочинять новые мелодии, которые приятно звучат и чем-то напоминают ранее известные, имеют хороший стиль и манеру, но в них всегда чего-то не хватает, чтобы стать действительно популярными. «Yesterday» и «Лунная соната» сочинены не компьютером.

Однако очень вероятно, что компьютер превзойдет человека и в этой области, и, как это ни обидно сознавать, в совсем недалеком будущем молодежь будет петь и танцевать под компьютерную музыку, а великие произведения, сочиненные классиками, будут считаться экзотикой.

6.3. Моделирование в поэзии

Считается, что задача моделирования стихотворчества несоизмеримо сложнее, чем задача моделирования сочинения музыкальных произведений. Как показали исследования русского языка, одна буква делового языка несет 0,6 бит информации, буква обыкновенной разговорной речи — 1 бит, а буква поэтической речи — 1,5 бит.

Не вдаваясь в философские и этические проблемы, подойдем к проблеме моделирования поэтического творчества с помощью известных инструментальных средств. Например, можно воспользоваться известным формализмом Бекуса — Наура. Этот формализм главным образом применяется в системах распознавания и обработки текстовой информации, машинного перевода, а также естественно-языкового общения. В формализме применяются следующие символы-операторы:

:: = — «определяется как» или «может быть переписан как»;

| (вертикальная черта) — используется для разделения различных альтернативных возможностей;

<> (угловые скобки) — используются для заключения нетерминальных символов, т. е. символов, которые должны определяться одним из правил. В отличие от них символы, не заключенные в угловые скобки, считаются терминальными и представляют лишь самих себя.

Например, цифра в формализме Бекуса — Наура определяется следующим образом:

<цифра>:: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9,

где *<цифра>* — нетерминальный символ, а каждая из цифр 0, 1, ..., 9 — терминальный символ.

Приведем некоторое множество правил упрощенного описания синтаксической структуры английского предложения.

1. *<предложение>*:: = *<существ. фрагмент> <глагольн. фрагмент> <существ. фрагмент>*

2. *<существ. фрагмент>*:: = *<артикль> <определяемое существ.>*

3. *<определяемое существ.>*:: = *< существ.>* | *<определение> <определяемое существ.>*

4. *<глагольн. фрагмент>*:: = *<глагол>* | *<глагольн. фрагмент> <фрагмент наречия>*

5. *<фрагмент наречия>*:: = *< наречие>* | *<предлог><существ. фрагмент>*

К этим правилам добавим небольшой словарь.

6. *<артикль>*:: = a | the

7. *<существ.>*:: = cat | mouse | dog | fox (кошка, мышь, собака, лиса)

8. *<определение>*:: = quick | lazy | brown | black (быстрый, ленивый, коричневый, черный)

9. *<глагол>*:: = goes | jumps | runs (идет, прыгает, бежит)

10. *<наречие>*:: = quickly | easily (быстро, легко)

11. *<предлог>*:: = over | under | through (над, под, через)

Приведенное множество синтаксических правил обычно используется для синтаксического анализа текстовой информации. Однако этот же фрагмент можно использовать вместе с генератором случайных чисел для создания случайных предложений, подчиня-

ющихся указанным синтаксическим правилам, т. е. можно запустить процедуру синтаксического анализа как бы в обратную сторону.

Чтобы построить какое-нибудь предложение, процесс генерации нужно запустить с правила 1. Здесь имеются две возможности, и для выбора какой-то одной из них следует обратиться к генератору псевдослучайных чисел. Затем процесс генерации развивается в соответствии с выбранной альтернативой, т.е. если выбрана первая из возможностей, то управление сначала должно перейти к правилу 2 для создания «существ. фрагмента», а после того, как это сделано, — к правилу 4 для создания «глагольн. фрагмента». Поскольку правила 2 и 4, в свою очередь, передают управление другим правилам, то необходимо иметь стек, или магазинную память, для напоминания о неоконченных моментах в различных правилах. Когда осуществляется первая передача управления от правила 1 к правилу 2, в стек помещается первый элемент и указатель — на следующий по порядку элемент (в соответствии с выбранной альтернативой).

Если в процессе генерации встречается терминальный символ, то он добавляется к выходной строке, которая создается.

Приведенный способ генерации предложений позволяет получить такие фразы, как: «The quick brown fox jumps over the lazy dog» — проворная коричневая лиса перепрыгнула через ленивую собаку.

Всего в словаре приведенного множества содержится 18 слов, выступающих в качестве терминальных символов: два артикля, четыре существительных, четыре прилагательных и т.д. Однако приведенный метод применим и к большим словарям, содержащим яркие, выразительные слова. Таким образом, можно заставить компьютер сочинять стихи, в которых, однако, не будет соблюдаться рифма. Для выдерживания рифмы необходимо введение дополнительных правил, рифмующих фразы.

Для того чтобы сочиненные компьютером произведения имели смысл, необходимо ввести базу соответствующих знаний. Каждое вводимое в словарь слово должно быть увязано с другими не только синтаксическими, но и семантическими связями. Различные сочетания слов должны быть оценены некоторыми оценивающими параметрами, задающими уровень смыслового соответствия. Такие параметры должны использоваться при построении фраз, подобно тому как в экспертных системах при получении заключений используются коэффициенты доверия. Этими коэффициентами можно регулировать уровень осмысленности и степень абстрагизма создаваемого произведения, определять его характер, жанр и смысловую направленность.

Как и в других интеллектуальных системах, коэффициенты доверия могут меняться в процессе работы программы, т.е. алгоритмы стихотворчества могут быть обучаемыми.

Таким образом, существующий уровень развития инструментальных средств и методов искусственного интеллекта позволяет создать более-менее приемлемые алгоритмы поэтического творчества, что свидетельствует о том, что и этот вид человеческой деятельности в принципе поддается компьютерному моделированию.

Контрольные вопросы

1. Перечислите общие принципы построения математических моделей.
2. Дайте определение инварианта и приведите примеры инвариантов, известных вам из математики, физики, искусства.
3. Сформулируйте теорему Геделя и поясните, какое отношение она имеет к творчеству?
4. Что такое творчество с точки зрения теории инвариантов?
5. Сформулируйте теорему Мак-Каллока—Питтса и поясните, какое отношение она имеет к творчеству.
6. Если бы вам предстояло писать программу компьютерного сочинения музыки, какую бы блок-схему вы предложили?
7. Поясните, каким образом можно использовать для сочинения стихов формализм Бекуса—Наура? Каким образом можно, пользуясь этим алгоритмом, придавать сочиненным произведениям смысл, изменять степень абстрагизма, определять его характер, жанр?

ГЛАВА 7

ИНТЕЛЛЕКТУАЛЬНОЕ МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

7.1. СОВРЕМЕННЫЙ КРИЗИС ПРИКЛАДНОЙ МАТЕМАТИКИ

Производя математические вычисления, мы пользуемся теми или иными формулами, которые, по существу, являются абстрактными моделями реальной действительности. Поэтому под термином «математическое моделирование» понимают все, что связано с практическим применением математики.

Развитие современной цивилизации свидетельствует о постоянном повышении роли математического компьютерного моделирования как в научных исследованиях, так и в различных областях практической деятельности человека. Одним из наиболее популярных инструментов математического компьютерного моделирования является формулировка и решение краевых задач математической физики. Именно таким способом ведутся многие фундаментальные исследования различных разделов физики, астрономии, экологии и других наук о человеке, природе и обществе, выполняются долгосрочные прогнозы погоды, предсказываются землетрясения, цунами и прочие стихийные бедствия, а также рассчитываются и проектируются самолеты, автомобили, ракеты, подводные лодки, здания и сооружения, различные промышленные и военные объекты.

В истории развития методов решения краевых задач математической физики можно проследить три периода. Первый исторический период, продлившийся примерно до середины XX в., начался с основополагающих работ Ж.Л.Д'Аламбера и Ж.Фурье, выполненных в XVIII—начале XIX вв. Путем разделения переменных удалось получить ряд решений дифференциальных уравнений в частных производных для простейших областей, называемых каноническими, — круга, квадрата, цилиндра, шара и пр. Затем на протяжении полутора веков усилия математиков сводились к развитию метода разделения переменных и изобретению иных приемов, позволяющих получить решение той или иной краевой задачи для других дифференциальных уравнений, других областей с другими краевыми условиями. Каждое такое решение было событием в математическом мире и отмечалось присуждением премий и присвоением регалий. Метод математического моделирования был доступен узкому кругу математиков-профессио-

оналов, деятельность которых представляла собой творческий процесс сродни деятельности поэтов, художников, композиторов.

Появление в середине XX в. быстродействующих электронно-вычислительных машин в корне изменило ситуацию. Оказалось, что если разбить область решения краевой задачи на множество мелких подобластей и для каждой подобласти ввести гипотезы, упрощающие физические свойства среды, то процесс интегрирования дифференциальных уравнений можно свести к множеству элементарных арифметических действий. Таким образом, краевые задачи математической физики стало возможным решать с помощью ЭВМ «с позиции грубой силы», получая решение не в виде аналитических формул, а в виде массивов чисел. Так появилась на свет новая область математики, называемая дискретной. На смену классическим аналитическим методам пришли численные алгоритмы, с помощью которых удалось создать универсальные пакеты прикладных программ, оснащенных удобными сервисными средствами. Математическое компьютерное моделирование стало общедоступным и из творчества превратилось в ремесло. Математики–аналитики с их хитроумными математическими выкладками, казалось, навсегда утратили свой авторитет и отошли в прошлое.

Однако, как утверждают философы, жизнь развивается по спирали. Маятник, качнувшийся в одну сторону, должен обязательно отклониться и в другую. Увлечение численными методами в полной мере выявило не только их бесспорные преимущества, но и неустранимые недостатки. К последним относится невозможность надежной оценки погрешности расчетных результатов. Этот недостаток особенно ощутим в последнее время в связи с применением метода математического моделирования для расчета ответственных объектов и процессов, от которых зависит безопасность людей, государств, цивилизации.

Следует заметить, что математический аппарат, которым пользовались математики минувших веков, был более надежен. Решения, полученные аналитическими методами в виде аналитических формул, могут быть проверены на удовлетворение дифференциальным уравнениям и краевым условиям решаемой задачи, т. е. их погрешность может быть оценена. Решения же, получаемые численными методами, представляют собой массивы чисел, о погрешности которых судят по тому, как эти числа изменяются с увеличением числа разбиений заданной области. Обычно считают, что результатам можно доверять, если они перестают изменяться с измельчением сетки. Однако уже давно показана теоретическая несостоятельность этого подхода. Дело в том, что с измельчением конечноэлементной сетки ухудшается обусловленность матрицы разрешающих алгебраических уравнений. Так, в случае решения двумерной краевой задачи для дифференциальных урав-

нений второго порядка и применения равномерной сетки с линейными функциями формы имеет место зависимость

$$\alpha = Ch^{-2}, \tag{7.1}$$

где α — спектральное число обусловленности матрицы системы алгебраических уравнений; C — константа, зависящая от задачи; h — максимальный размер элемента.

Согласно этой формуле при уменьшении h увеличивается α, т.е. коэффициенты матрицы системы алгебраических уравнений хуже обусловливают ее решение: малые изменения коэффициентов матрицы начинают приводить к большим изменениям решения системы. Это значит, что погрешности, связанные, например, с округлением коэффициентов матрицы или вносимые в эти коэффициенты в процессе их формирования, все сильнее и сильнее влияют на результат решения системы. А это, в свою очередь, означает, что при $h \to 0$ приближенные конечноэлементные решения сходятся вовсе не к искомому точному решению краевой задачи, как схематично показано на рис. 7.1.

Из приведенного анализа со всей очевидностью следует, что к результатам, полученным численными методами, следует относиться крайне осторожно, особенно, если речь идет о расчетах объектов и процессов ответственного назначения. Тем не менее, на современном рынке программных средств имеется множество компьютерных программ, реализующих численные методы решения краевых задач теплопроводности, гидродинамики, теории упругости, теории электрических, магнитных, гравитационных и даже торсионных полей. Эти пакеты (ANSYS, KOSMOS, WINMASHIN и др.) снабжены удобными сервисными и графическими средствами, так что любой пользователь, далекий от математики, может без особого труда получить приемлемое с точки зрения здравого смысла приближенное решение практически любой краевой задачи. Однако оценить, на сколько полученное им решение отличается от настоящего точного решения краевой задачи, представляет большую проблему. Понимая это, авторы численных пакетов в программной документации обычно указывают на то, что

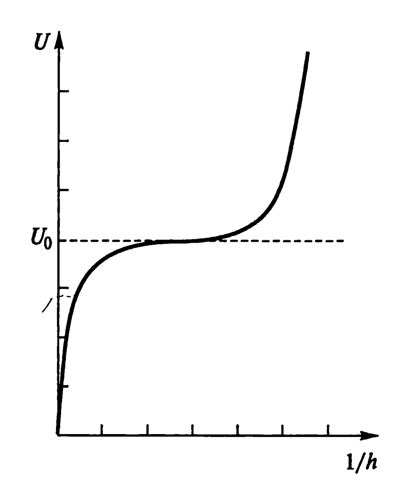


Рис. 7.1. Характерная зависимость численного решения краевой задачи от максимального размера конечноэлементарной сетки h:
U_0 — точное решение задачи

разработчики программ не несут ответственности за последствия выполненных расчетов.

По прогнозам специалистов, XXI в. — это век жестоких техногенных катастроф, стихийных и экологических бедствий. Малейшие ошибки в математических моделях и, в частности, в методах решения краевых задач, могут привести к тяжелым экологическим, экономическим и социальным последствиям. Все чаще приходится слышать сообщения о падении ракет, самолетов, взрывах на ядерных станциях и промышленных объектах, обрушениях жилых зданий и развлекательных комплексов. В связи с этим, как никогда прежде, стали актуальными вопросы точности математического моделирования. Пришло время применять только такие модели и методы, которые могут гарантировать необходимую надежность расчетных результатов.

Надежность результатов математического моделирования обеспечивают точные аналитические методы, которые разрабатывали математики прошлых веков. На современном рынке программной продукции программных пакетов, реализующих такие методы, практически нет. Их нет потому, что аналитические методы не универсальны и крайне плохо поддаются алгоритмизации. Они требуют творческого интеллектуального подхода практически к каждой новой краевой задаче. Их применение связано с длительной и кропотливой работой высокообразованных математиков, использующих весь свой опыт, интуицию, талант. Традиции же школы математиков–аналитиков, создававшейся на протяжении нескольких последних веков, к сожалению, в значительной степени утеряны.

Выход из кризисной ситуации связывают с новым (третьим) периодом развития методов решения краевых задач, наметившимся в связи с очередной революцией в компьютерной индустрии — интеллектуализацией компьютеров, развитием и внедрением идей и методов искусственного интеллекта. Компьютерная имитация творческой деятельности человека (эксперта), его интуиции и опыта позволила создать и успешно применять компьютерные программы в различных сферах человеческой деятельности, ранее считавшихся недоступными для формализации и алгоритмизации, таких как медицина, политология, социология, финансы, бизнес и т.д. Не является исключением применение идей и методов искусственного интеллекта и при решении краевых задач математической физики.

Попытки имитации творческой деятельности математика-аналитика показали перспективность нового подхода, принципиальную возможность реанимации старых аналитических методов и создания на этой основе универсальных программных комплексов, способных получать надежные аналитические решения краевых задач, пригодные для моделирования объектов и процессов ответственного назначения.

На сайте http://www.pspu.ru/regions размещен доступный для общего пользования программный пакет REGIONS, который, по существу, является экспертной системой, имитирующей творческую деятельность математика-профессионала (эксперта), выполняющего решение краевой задачи. В основу пакета положен малоизвестный аналитический метод — метод фиктивных канонических областей (ФКО) [49—52]. Этот метод, предложенный автором книги в начале 70-х годов, хотя и позволил найти точные аналитические решения ряда практически важных задач, не нашел широкого применения ввиду отсутствия универсальности (что, кстати, свойственно всем аналитическим методам). Дело в том, что успех решения краевой задачи методом ФКО в значительной степени зависит от опыта и интуиции применяющего его математика. Теперь же появилась возможность заложить в компьютер интеллект математика-профессионала в виде набора эвристических правил. Эти правила в большинстве случаев не имеют строгих математических доказательств и поэтому обладают различными *коэффициентами доверия*. Мы имеем дело с *нечеткой математикой*, характерной для систем искусственного интеллекта. Результат же применения нового подхода — точное аналитическое решение краевой задачи, не нуждающееся в оценке погрешности и не вызывающее сомнений в своей надежности. Такой подход к решению краевых задач назван *интеллектуальным компьютерным математическим моделированием*.

7.2. МЕТОД ФИКТИВНЫХ КАНОНИЧЕСКИХ ОБЛАСТЕЙ

7.2.1. Идея и теоретические основы

В 1926 г. Е. Треффтцем был предложен аналитический метод решения краевых задач, суть которого заключается в следующем.

Пусть требуется решить уравнение Лапласа

$$\Delta U(p) = 0 \tag{7.2}$$

в некоторой области D, на границе S которой задано условие

$$U(p)|_S = U^*(S). \tag{7.3}$$

Согласно методу Треффтца решение краевой задачи ищется в виде разложения

$$U(p) = \sum_{n=1}^{N} c_n U_n(p), \ p \in D, \tag{7.4}$$

в котором $U_n(p)$ — базисные функции координат, выбираемые так, что каждая из них удовлетворяет заданному дифференциаль-

ному уравнению (7.2), а c_n — постоянные коэффициенты, определяемые из условия минимума функционала

$$J(U) = \int_D (gradU(p))^2 dD, \tag{7.5}$$

соответствующего краевому условию (7.3).

Таким образом, сумма (7.4) тождественно удовлетворяет заданному дифференциальному уравнению и приближенно — краевому условию. Это свойство решений, получаемых методом Треффтца, открывает уникальную возможность надежной оценки точности результатов. В самом деле, после того, как краевая задача решена и постоянные c_n определены, мы имеем возможность подставить их в сумму (7.4) и подсчитать ее значение на границе S заданной области D. Таким образом, вместо условия (7.3) мы имеем другое условие:

$$U(p)\big|_S = \tilde{U}^*(S), \tag{7.6}$$

в котором граничное значение найденного решения $\tilde{U}^*(S)$ отличается от заданного $U^*(S)$ на некоторую величину $\delta(S) = \left|U^*(S) - \tilde{U}^*(S)\right|$, причем максимальное значение этой разности в каждом конкретном случае может быть легко подсчитано.

Итак, вместо решения заданной краевой задачи (7.2), (7.3) метод Треффтца позволяет получить точное аналитическое решение краевой задачи (7.2), (7.6), граничное значение искомой функции которой отличается на величину $\delta(S)$.

В практике компьютерного моделирования некоторая корректировка граничных условий краевых задач, как правило, допускается. Дело в том, что граничные условия обычно формулируются с привлечением различных физических гипотез и допущений либо являются результатом измерений физических приборов, которые всегда имеют некоторую погрешность ε, и если

$$\max_S \delta(S) < \varepsilon, \tag{7.7}$$

то исходную краевую задачу (7.2), (7.3) можно заменить на задачу (7.2), (7.6). Но эту задачу, как мы только что видели, методом Треффтца удается решить точно. Таким образом, вопрос об оценке погрешности решения краевой задачи, полученного методом Треффтца, снимается в принципе.

Это замечательное свойство метода Треффтца ставит его в особое положение по отношению к другим приближенным подходам, является его серьезнейшим преимуществом, особенно важным в современных условиях, когда на первое место выходят качество и надежность результатов.

Подход, предложенный Е. Треффтцем, в свое время вызвал серию научных работ, посвященных вопросам его развития и при-

менения. Среди них следует отметить фундаментальные работы Э. Рейснера, Л. С. Лейбензона, С. Г. Михлина, М. Ш. Бирмана, Г. А. Гринберга, Л. Коллатца. Однако, несмотря на уникальные свойства и внимание математиков, метод Треффтца долгое время оставался не пригодным для широкого практического применения. Дело в том, что нерешенной была проблема выбора базисных функций $U_n(p)$, удовлетворяющих решаемым дифференциальным уравнениям и обеспечивающих сходимость метода. Только в редких случаях путем увеличения числа функций N удавалось уменьшить до приемлемых значений погрешность удовлетворения краевым условиям $\delta(S)$ и получить более-менее приемлемые решения краевых задач. Таким образом, успех применения метода целиком и полностью зависел от опыта и интуиции математика, а порой и просто от везения.

Разобраться в проблемах сходимости и корректности, построить методику выбора базисных функций, обеспечивающую успех применения метода Треффтца, позволила предложенная в 1973 г. Л. Н. Ясницким [49] геометрическая интерпретация процесса решения краевых задач, суть которой поясним на примере краевой задачи теории упругости.

Пусть требуется рассчитать напряженно-деформированное состояние упругого тела D, изображенного на рис. 7.2, *а*. На поверхности S тела D заданы граничные условия в перемещениях или (и) в напряжениях. Наряду с D в рассмотрение вводится некоторая фиктивная каноническая область V, внутри которой мысленно (на рис. 7.2, *б* — пунктиром) выделяются контуры заданного тела. Поскольку область V является канонической, то для нее методом разделения переменных Фурье можно построить решение дифференциальных уравнений теории упругости, имеющее вид ряда

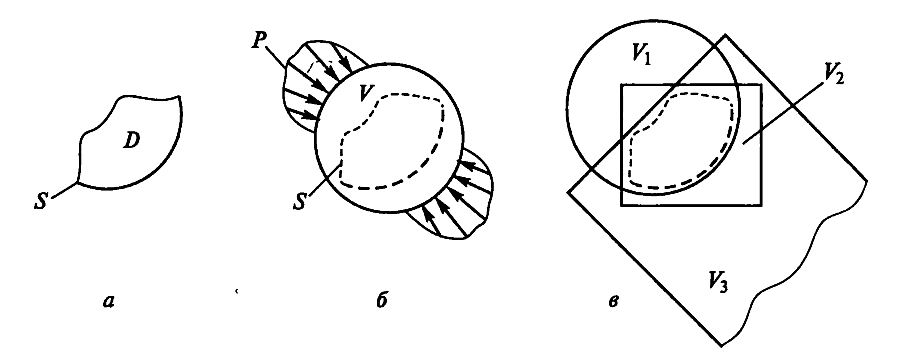


Рис. 7.2. Заданное тело D (*а*) мысленно погружается в каноническую область V (*б*) или область пересечения нескольких канонических областей $V_1 \cap V_2 \cap V_3$ (*в*)

$$U(p)=\sum_{n=1}^{N}c_nU_n(p),\quad N\to\infty,\quad p\in V,\tag{7.8}$$

где c_n — постоянные коэффициенты, определяемые граничными условиями на поверхности области V; $U_n(p)$ — координатные функции, тождественно удовлетворяющие решаемым дифференциальным уравнениям.

Это решение является общим в том смысле, что путем подбора постоянных коэффициентов c_n из него могут быть выделены частные решения, удовлетворяющие достаточно произвольным граничным условиям на поверхности области V. Если теперь на этой поверхности создать такое нагружение P, что на контурах вписанного тела D возникнут напряжения или (и) перемещения, совпадающие с заданными на S граничными условиями, то решение для V, соответствующее нагружению P, будет в то же время являться решением исходной задачи для тела D. Последнее справедливо в силу того, что выделенное из (7.8) частное решение удовлетворяет внутри тела D уравнениям теории упругости, а на его поверхности — заданным граничным условиям.

С математической точки зрения задача нагружения фиктивного тела V состоит в определении коэффициентов разложения (7.8), обеспечивающих выполнение заданных на S условий. Если ограничиться конечным числом слагаемых N ряда (7.8), то эту задачу можно решить приближенно, удовлетворяя граничным условиям в N точках коллокаций, лежащих на S, или минимизируя на этой поверхности функционал граничных условий. Последний может быть сформулирован по методу Треффтца (7.5) с использованием энергетических представлений либо по методу наименьших квадратов (см. подразд. 7.2.3).

Приведенная здесь геометрическая интерпретация к математическому аппарату Треффтца — Рейсснера — Лейбензона — Михлина позволила проанализировать условия сходимости и корректности, сформулировать и доказать соответствующие теоремы, выработать критерий выбора фиктивных канонических областей, обеспечивающих успех решения задач, и дать рекомендации по практическому выполнению этого критерия. В результате была предложена методика выбора базисных функций к известному математическому аппарату, обеспечивающая успех применения этого аппарата и названная впоследствии методом фиктивных канонических областей (ФКО) [50].

В настоящее время метод ФКО используется в базе знаний интеллектуальной системы математического моделирования REGIONS, предназначенной для получения аналитических решений краевых задач математической физики.

Суть критерия выбора ФКО заключается в требовании продолжимости в V искомого решения как функции, удовлетворяющей

дифференциальным уравнениям задачи, причем под V подразумевается минимальная содержащая D область из семейства областей, для которых имеет место используемое разложение (7.8). В качестве V может быть выбрана как отдельная каноническая область (см. рис. 7.2, *б*), так и область пересечения нескольких канонических областей $V = V_1 \cap V_2 \cap ... \cap V_K$ (рис. 7.2, *в*). В этом случае вместо (7.8) используется сумма K разложений, относящихся соответственно к V_1, V_2, ..., V_K.

Критерий выбора ФКО основан на том, что при выполнении условия продолжимости искомого решения в V это решение является суммой по базисным функциям для V. Тогда разложение (7.8) (или сумма K разложений в случае суперпозиции фиктивных областей) при ограниченном числе N представляет собой отрезок ряда и для него применимы соответствующие теоремы сходимости. Если же продолжимость отсутствует, то такого ряда не существует. Сумма же в правой части формулы (7.8) может рассматриваться только как линейная комбинация функций $U_n(p)$, аппроксимирующая в D искомое решение. Но для такой аппроксимации задача нахождения коэффициентов c_n не является корректной по Адамару. В этом случае, как доказано С. Я. Гусманом [8], при увеличении N некоторые из коэффициентов c_n неограниченно возрастают. Это значит, что при практических вычислениях на компьютерах в принципе невозможно получить решение задачи с погрешностью, меньшей некоторого положительного числа ε_0, зависящего от краевой задачи и памяти компьютера.

Следует отметить, что ε_0 может оказаться достаточно малым, и поэтому выполнение критерия продолжимости не является строго обязательным. Однако сам факт существования ε_0, ограничивающего точность решения задачи снизу, является нежелательным и, как правило, не позволяет получать приемлемых результатов. Поэтому в методе ФКО выполнению условия продолжимости придается первостепенное значение.

Нарушение продолжимости может происходить из-за наличия особенностей, т. е. точек, в которых искомое решение обращается в бесконечность, имеет разрывы, изломы и т. п. В реальных краевых задачах особые точки решений, как правило, располагаются за пределами тела D либо на его поверхности. Таким образом, задача выбора ФКО сводится к тому, чтобы:

предсказать возможные места расположения особых точек;

подобрать и расположить ФКО так, чтобы особые точки искомого решения лежали за пределами области V (или на ее поверхности).

Продемонстрируем применение критерия выбора ФКО для решения следующих задач.

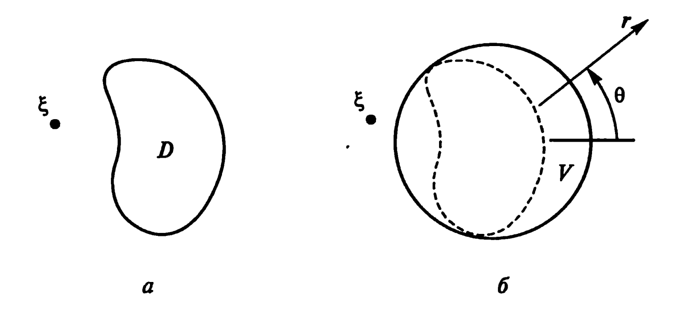


Рис. 7.3. Заданное тело D (*а*) погружается в круг V (*б*):
ξ — особая точка искомого решения

Задача 1. Решить уравнение Лапласа для плоского тела D, изображенного на рис. 7.3, *а*. Известно, что искомое решение имеет особую точку ξ, расположенную вблизи D.

В качестве фиктивной области V в этом случае можно использовать, например, круг, для которого имеет место решение уравнения Лапласа

$$U = \sum_{n=0}^{N} r^n (a_n \cos n\theta + b_n \sin n\theta), \quad N \to \infty, \tag{7.9}$$

где r, θ — полярные координаты; a_n и b_n — постоянные коэффициенты.

Согласно критерию продолжимости круг надо расположить так, чтобы он содержал D и не содержал ξ. Пример такого расположения показан на рис. 7.3, *б*.

Задача 2. На рис. 7.4, *а* приведен случай, когда подобрать круг, удовлетворяющий критерию продолжимости, не удается. Поэтому в качестве V предложена кольцевая область (рис. 7.4, *б*), для которой имеет место разложение

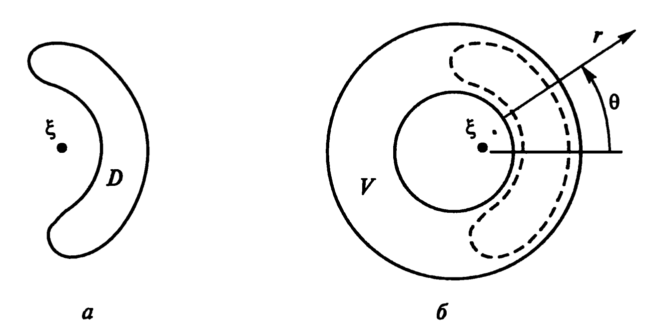


Рис. 7.4. Заданное тело D (*а*) погружается в кольцевую область V (*б*)

$$U = \sum_{n=-N}^{N} r^n (a_n \cos n\theta + b_n \sin n\theta) + c_0 \ln r, \quad N \to \infty. \tag{7.10}$$

Задача 3. На рис. 7.5, *а* приведен более сложный случай, когда вблизи тела D имеются две особые точки: ξ_1 и ξ_2. Ни круг, ни кольцо для этой краевой задачи не годятся, поэтому здесь рекомендуется прием суперпозиции ФКО [51]: D погружается в область пересечения двух кольцевых областей $V = V_1 \cap V_2$. Разложение для V представляет собой сумму двух рядов

$$U = U_1 + U_2, \tag{7.11}$$

первый из которых

$$U_1 = \sum_{n=-N}^{N} r_1^{\,n} (a_{n1} \cos n\theta_1 + b_{n1} \sin n\theta_1) + c_1 \ln r_1, \quad N \to \infty \tag{7.12}$$

относится к области V_1, а второй

$$U_2 = \sum_{n=-N}^{N} r_2^{\,n} (a_{n2} \cos n\theta_2 + b_{n2} \sin n\theta_2) + c_2 \ln r_2, \quad N \to \infty \tag{7.13}$$

к области V_2.

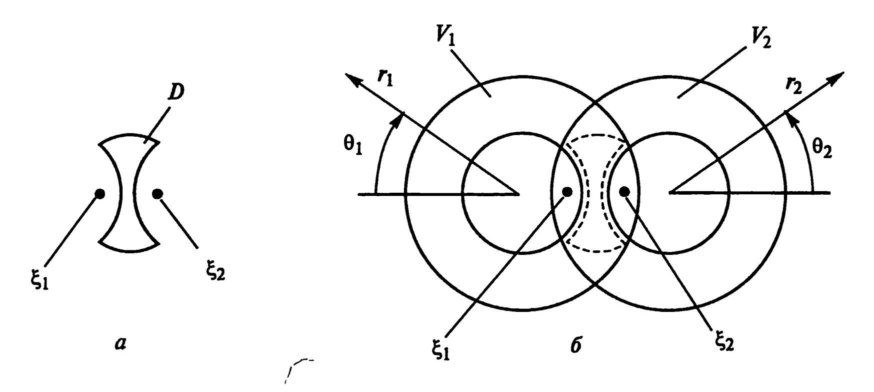


Рис. 7.5. Заданное тело D (*а*) погружается в область пересечения двух фиктивных колец V_1 и V_2 (*б*)

Как видно из рисунка 7.5, *б*, особые точки ξ_1 и ξ_2 оказались за пределами области пересечения фиктивных колец и искомое решение, таким образом, продолжимо в $V = V_1 \cap V_2$.

Задача 4. На рис. 7.6 приведен вариант применения приема [52], названного композицией расчетной области. Заданное тело D разбивается на три элемента D_1, D_2 и D_3, между которыми задаются условия совместности перемещений и напряжений. Затем каждый из элементов погружается в соответствующую фиктивную кано-

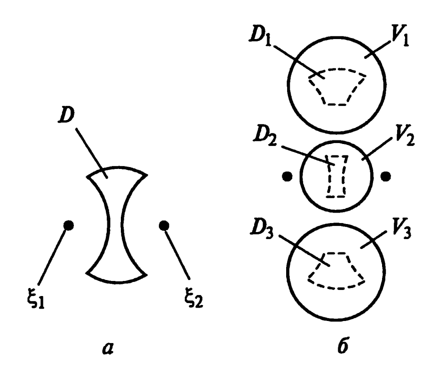


Рис. 7.6. Заданное тело D (a) расчленяется на элементы D_1, D_2, D_3, которые погружаются в канонические области V_1, V_2, V_3 ($б$)

ническую область V_1, V_2 и V_3 с таким расчетом, чтобы особые точки ξ_1 и ξ_2 оказались за их пределами. Искомое решение задачи для D представляется в виде совокупности разложений, каждое из которых относится к своей канонической области:

$$U_m = \sum_{n=0}^{N} r_m{}^n (a_{nm} \cos n\theta_m + b_{nm} \sin n\theta_m),$$

$$r_m, \theta_m \in D_m, \ m = 1, 2, 3, \ N \to \infty. \qquad (7.14)$$

В практических задачах часто удается сделать так, чтобы подобласти D_m сами по себе были каноническими областями, совпадающими по форме с V_m. Такой способ разбиения и выбора канонических областей приводит к получению наиболее эффективных, быстро сходящихся алгоритмов.

Как показал опыт, применение приемов суперпозиции и композиции позволяет всегда добиться выполнения условия продолжимости. Поэтому методом ФКО можно в принципе решать линейные краевые задачи любой степени сложности.

7.2.2. Иллюстрации на тестовой задаче и другие правила

Представляет интерес применить метод ФКО к задаче, решение которой заранее известно. Пусть требуется решить уравнение Лапласа в плоском теле, ограниченном двумя конфокальными эллипсами (рис. 7.7), оси которых определены формулами

$$\begin{aligned} A_1 &= C\text{ch}\gamma_1; \quad B_1 = C\text{sh}\gamma_1; \\ A_2 &= C\text{ch}\gamma_2; \quad B_2 = C\text{sh}\gamma_2, \end{aligned} \qquad (7.15)$$

где $C = \text{const}$ — расстояние между фокусами; $0 < \gamma_1 < \gamma_2 < \infty$.

На внутреннем эллипсе задано значение функции U_1, а на внешнем — значение функции U_2.

Для этой задачи известно точное аналитическое решение:

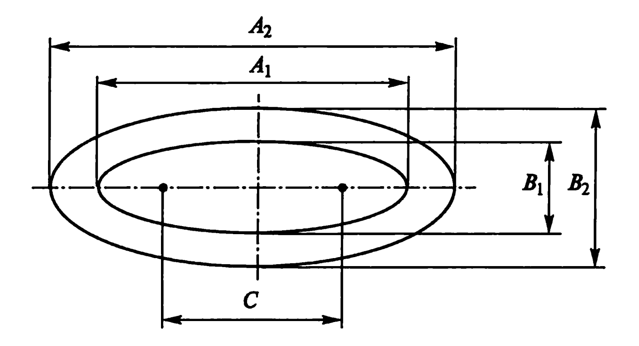


Рис. 7.7. Плоское тело, ограниченное конфокальными эллипсами

Рис. 7.8. Погружение четырех вариантов эллипсоидного тела D, оси которого заданы значениями табл. 7.1, в фиктивную кольцевую область V (слева) и соответствующие кривые сходимости (справа). Номер кривой соответствует номеру варианта. Для 1-го и 2-го вариантов условие продолжимости выполняется, для 3-го и 4-го — не выполняется

$$U = U_2 + \frac{\gamma_2 - \gamma}{\gamma_2 - \gamma_1}(U_1 - U_2). \qquad (7.16)$$

Заметим, что решение (7.16) имеет особенность типа излом (разрыв первой производной) на отрезке, соединяющем фокусы эллипсов (где $\gamma = 0$).

В качестве фиктивной канонической области для рассматриваемой задачи выберем кольцо, для которого имеет место разложение (7.10). Для ограниченного N постоянные a_n, b_n, c_0 определяются из условия приближенного удовлетворения граничным условиям задачи, например методом наименьших квадратов (см. подразд. 7.2.3). Результаты вычислений представлены на рис. 7.8 в виде зависимостей максимальной погрешности решения краевой задачи ε от числа удержанных слагаемых в разложении (7.10) $m = 2N + 1$. Эти зависимости построены для различных соотношений между осями эллипсов (табл. 7.1). Как видно из рис. 7.8, поведение кривых существенно зависит от геометрических параметров эллиптических областей. Хорошая сходимость наблюдается для 1-го и 2-го вариантов, для которых фокусное расстояние C меньше меньшей оси внутреннего эллипса B_1. Поскольку ось B_1 совпадает с диаметром внутренней окружности фиктивного кольца, а особенность искомого решения расположена на отрезке, соединяющем фокусы эллипсов, то при $C < B_1$ эта особенность находится за пределами фиктивной канонической области и не препятствует выполнению условия продолжимости. Для 3-го и 4-го вариантов $C > B_1$, условие продолжимости не выполняется и, как зафиксировано на рис. 7.8, приближенные решения не сходятся к точному.

Таким образом, численные эксперименты подтверждают справедливость критерия выбора фиктивных канонических областей: если условие продолжимости выполняется, то приближенные решения сходятся к точному; если же это условие не выполняется, то и сходимость отсутствует.

Таблица 7.1.

Варианты параметров эллипсоидального тела

Номер варианта (номер кривой на рис. 7.8)	Бóльшая ось внутреннего эллипса A_1	Меньшая ось внутреннего эллипса B_1	Бóльшая ось внешнего эллипса A_2	Меньшая ось внешнего эллипса B_2	Фокусное расстояние C	Соотношение между C и B_1
1	1,05	0,990	1,2	1,1	0,478	$C < B_1$
2	1,05	0,816	1,2	1,0	0,660	$C < B_1$
3	1,05	0,550	1,2	0,8	0,900	$C > B_1$
4	1,05	0,390	1,2	0,7	0,980	$C > B_1$

Обсуждая возможность решения вариантов рассмотренной модельной задачи, для которых $C > B_1$, укажем на необходимость использования вместо кольца других типов фиктивных канонических областей, удовлетворяющих критерию продолжимости. Однако проще воспользоваться приемом суперпозиции базисных разложений, относящихся к двум фиктивным кольцам, центры которых смещены на расстояние H друг от друга, как показано на рис. 7.9. Здесь же построены кривые сходимости приближенных решений 3-го варианта задачи (см. табл. 7.1) при варьировании величины H. Как видно из рисунка, только в одном случае, когда $H = 0{,}44$, приближенные решения сходятся к точному. Именно этот случай и соответствует выполнению критерия продолжимости: особенность искомого решения здесь полностью исключена из области пересечения фиктивных кольцевых областей $V = V_1 \cap V_2$. В других трех случаях ($H = 0$; $H = 0{,}2$; $H = 0{,}9$) критерий продолжимости не выполняется и, как следствие, сходимость решений отсутствует.

Таким образом, проведенные вычислительные эксперименты подтверждают справедливость критерия продолжимости как в случае использования одной фиктивной канонической области, так и в случае суперпозиции нескольких фиктивных канонических областей.

Как уже упоминалось выше, теоретически возможны случаи получения приемлемых решений и тогда, когда условие продолжимости не выполняется. Опыт решения краевых задач методом ФКО показал, что такие исключения из общего правила встречаются крайне редко, а сами получаемые решения в таких случаях не отличаются высоким качеством. Поэтому на языке искусственного интеллекта критерий продолжимости можно квалифицировать как эвристическое правило, обладающее высоким коэффициентом доверия.

Помимо критерия продолжимости можно сформулировать еще несколько более-менее обоснованных правил выбора ФКО. Так, целесообразно стремиться к тому, чтобы особенности искомого решения и используемого общего решения для канонической области были совмещены в пространстве. Ясно, что при таком совмещении автоматически будет выполняться и условие продолжимости.

Однако надо иметь в виду, что условие продолжимости является более общим и может выполняться и при несовмещенных особенностях, т.е. между координатами особенностей искомого решения для D и используемого общего решения для V допускается некоторое расстояние δ.

Следующий предложенный критерий заключается в требовании метрической близости областей D и V. Это требование непос-

редственно вытекает из геометрической интерпретации метода ФКО. Как ясно из физических соображений, добиться нужных напряжений (перемещений) на контурах заданного тела D тем легче, чем ближе между собой расположены в пространстве поверхности фиктивного и вписанного в него тела.

Еще один критерий — условие топологической эквивалентности (одинаковой степени связности) между областями D и V — предложен С. Ю. Большаковым и В. А. Елтышевым [2]. Этот критерий не всегда соответствует практическим результатам. Например, для случая, иллюстрируемого рис. 7.8, области D и V топологически эквивалентны во всех четырех вариантах, между тем как сходимость приближенных решений к точному наблюдается толь-

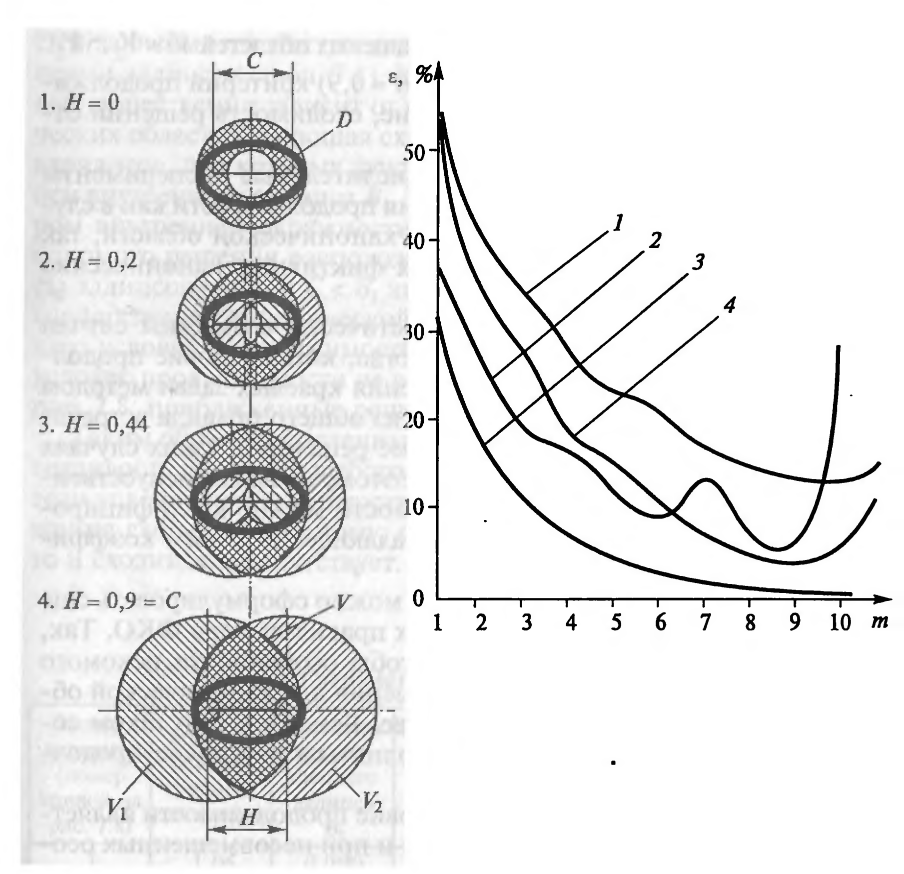


Рис. 7.9. Случаи погружения эллипсоидального тела D, оси которого заданы вариантом 3 табл. 7.1, в область пересечения двух фиктивных колец $V_1 \cap V_2$ (слева) и соответствующие кривые сходимости (справа). Номер кривой соответствует номеру случая погружения. Условие продолжимости выполняется только для 3-го случая погружения

ко в 1-м и 2-м вариантах. Для случая, показанного на рис. 7.9, условие топологической эквивалентности между D и $V = V_1 \cap V_2$ выполняется в 1, 2 и 3-м вариантах, тогда как сходимость зафиксирована только в 3-м варианте.

Вместе с тем есть основания предположить, что требование топологической эквивалентности между D и V для определенных классов задач может оказаться полезным. Тем более, что теория построения экспертных систем допускает использование как нечетких, так и неверно сформулированных правил, поскольку вердикт о полезности правила выносится окончательным значением соответствующего коэффициента доверия, вычисленным в результате обучения.

Итак, мы привели четыре правила выбора ФКО. Наиболее математически обоснованным и практически подтвержденным является критерий продолжимости. Использование других трех критериев может оказаться полезным для определенных классов задач, поэтому их также имеет смысл включать в базы знаний интеллектуальных систем, задавая этим правилам невысокие коэффициенты доверия либо предоставляя возможность системам самим определять коэффициенты доверия в процессе самообучения.

Для применения первых двух критериев необходим предварительный прогноз особенностей искомых решений. Математики, имеющие опыт решения краевых задач, обычно могут делать такой прогноз интуитивно. Однако в программных пакетах, предназначенных для широкого круга пользователей, рассчитывать на интуицию последних не приходится. В связи с этим в подразд. 7.3.1 изложены принципы создания алгоритма, имитирующего интуицию математика-профессионала, выполняющего прогноз особых точек искомых решений краевых задач.

7.2.3. Способы удовлетворения краевым условиям

После того как фиктивные канонические области выбраны и построены соответствующие им общие решения дифференциальных уравнений, решается задача определения коэффициентов, входящих в эти решения. Постоянные коэффициенты определяются из условий на поверхности заданного тела, причем задача их нахождения может решаться несколькими способами. Рассмотрим некоторые из них.

Метод граничной коллокации. Пусть требуется решить какое-либо дифференциальное уравнение для тела D, на границе S которого задано условие

$$U(p)\big|_S = U^*(S). \qquad (7.17)$$

Пусть в качестве фиктивной выбрана некоторая область V, для которой имеет место общее решение заданного дифференциального уравнения

$$U(p)=\sum_{n=1}^{\infty}c_nU_n(p),\quad p\in V. \tag{7.18}$$

Согласно методу граничной коллокации количество слагаемых в этом решении ограничивается конечным числом N:

$$U(p)=\sum_{n=1}^{N}c_nU_n(p),\quad p\in V. \tag{7.19}$$

На границе S заданного тела D через равные интервалы наносятся точки S_i, называемые точками коллокаций. Их число выбирается равным числу слагаемых, удержанных в общем решении (7.19). Далее в это решение подставляются координаты первой коллокационной точки и полученная сумма приравнивается к значению граничной функции (7.17) в этой точке:

$$\sum_{n=1}^{N}c_nU_n(S_1)=U^*(S_1). \tag{7.20}$$

Поочередный перебор всех точек коллокаций приводит к системе N линейных алгебраических уравнений относительно c_n:

$$\sum_{n=1}^{N}c_nU_n(S_i)=U^*(S_i),\ \ i=1,\ 2,\ ...,\ N. \tag{7.21}$$

Нахождение коэффициентов c_n из этой системы позволяет получить решение граничной задачи, которое удовлетворяет граничным условиям во всех коллокационных точках. Если выбор фиктивных областей сделан правильно, то погрешность удовлетворения граничным условиям на поверхности S между точками коллокаций будет уменьшаться с увеличением числа N.

Метод граничной коллокации достаточно универсален. В тех случаях, когда граничные условия накладываются не только на значения искомой функции, но и на их производные, координаты коллокационных точек вместо (7.17) подставляются в соответствующие граничные уравнения, что приводит к аналогичным системам алгебраических уравнений для определения постоянных коэффициентов базисных разложений.

Недостаток метода коллокаций состоит в том, что погрешность удовлетворения граничным условиям на поверхности S распределяется неравномерно.

Граничный метод наименьших квадратов. Согласно этому методу формируется функционал граничных условий, представляющий собой интеграл по границе области от квадратичной невязки удовлетворения граничным условиям:

$$J(U) = \int_S \left[\sum_{n=1}^{N} c_n U_n(p) - U^*(S) \right]^2 dS. \qquad (7.22)$$

Дифференцирование функционала по неопределенным коэффициентам c_k ($k = 1, 2, ..., N$) и приравнивание к нулю полученных производных приводит к системе линейных алгебраических уравнений

$$\sum_{n=1}^{N} c_n \int_S U_n(p)\, U_k(p)\, dS = \int_S U^*(S)\, U_k(p)\, dS, \quad k = 1, 2, ..., N, \qquad (7.23)$$

решение которой позволяет получить коэффициенты c_n, обеспечивающие выполнение граничных условий на S. Интегралы, входящие в (7.23), можно вычислять различными численными методами. Однако замечено, что наиболее простые и эффективные алгоритмы получаются, когда для вычисления интегралов используется формула средних прямоугольников, причем число прямоугольников берется в 5—7 раз больше порядка системы N. Более высокая точность вычисления интегралов, связанная с дополнительными затратами машинного времени, как правило, не приводит к улучшению решения краевой задачи в целом. При уменьшении числа прямоугольников до N метод наименьших квадратов вырождается в метод граничной коллокации.

Если граничные условия задачи имеют более сложный вид, чем условие (7.17), то метод наименьших квадратов нуждается в обобщении. Рассмотрим обобщение этого метода, предложенное Л. С. Лейбензоном [25].

Пусть надо решить задачу теории упругости в системе декартовых координат x, y, z для тела D, на части поверхности S_P которого заданы напряжения P_x^*, P_y^*, P_z^*. Пусть удалось подобрать фиктивную каноническую область V, для которой имеет место общее решение уравнений Ламе:

$$U_x = \sum_{n=1}^{N} c_n U_{xn}; \quad U_y = \sum_{n=1}^{N} c_n U_{yn}; \quad U_z = \sum_{n=1}^{N} c_n U_{zn}, \qquad (7.24)$$

где U_x, U_y, U_z — компоненты упругого смещения; U_{xn}, U_{yn}, U_{zn} — функции координат.

Подставив разложения (7.24) в выражение закона Гука, получим аналогичные разложения для компонент напряжений, действующих на поверхности S_P:

$$P_x = \sum_{n=1}^{N} c_n P_{xn}; \quad P_y = \sum_{n=1}^{N} c_n P_{yn}; \quad P_z = \sum_{n=1}^{N} c_n P_{zn}. \qquad (7.25)$$

Тогда суммарная квадратичная погрешность удовлетворения граничным условиям может быть представлена в виде интеграла

$$\varepsilon_P=\int\limits_{S_P}\left[\left(\sum_{n=1}^{N}c_nP_{xn}-P_x^*\right)^2+\left(\sum_{n=1}^{N}c_nP_{yn}-P_y^*\right)^2+\left(\sum_{n=1}^{N}c_nP_{zn}-P_z^*\right)^2\right]dS_P, \quad (7.26)$$

условие минимума которого ($\partial\varepsilon_P/\partial c_k = 0$, $k = 1, 2, ..., N$) эквивалентно системе линейных алгебраических уравнений относительно c_n:

$$\sum_{n=1}^{N}c_n\int\limits_{S_P}\left(P_{xn}P_{xk}+P_{yn}P_{yk}+P_{zn}P_{zk}\right)dS_P =$$
$$=\int\limits_{S_P}\left(P_x^*P_{xk}+P_y^*P_{yk}+P_z^*P_{zk}\right)dS_P. \quad (7.27)$$

Если на части поверхности S_U тела D заданы перемещения U_x^*, U_y^*, U_z^*, то функционал граничных условий имеет аналогичный вид:

$$\varepsilon_U=\int\limits_{S_U}\left[\left(\sum_{n=1}^{N}c_nU_{xn}-U_x^*\right)^2+\left(\sum_{n=1}^{N}c_nU_{yn}-U_y^*\right)^2+\left(\sum_{n=1}^{N}c_nU_{zn}-U_z^*\right)^2\right]dS_U, \quad (7.28)$$

а его минимальное значение обеспечивает система алгебраических уравнений

$$\sum_{n=1}^{N}c_n\int\limits_{S_U}\left(U_{xn}U_{xk}+U_{yn}U_{yk}+U_{zn}U_{zk}\right)dS_U =$$
$$=\int\limits_{S_U}\left(U_x^*U_{xk}+U_y^*U_{yk}+U_z^*U_{zk}\right)dS_U. \quad (7.29)$$

В технике наиболее часто встречаются краевые задачи со смешанными граничными условиями, когда на поверхности упругого тела заданы граничные условия как в напряжениях, так и в перемещениях. Для таких задач удобно использовать обобщение методики Л. С. Лейбензона. Формируется обобщенный функционал

$$\varepsilon = \frac{1}{S_P}\varepsilon_P + k\frac{1}{S_U}\varepsilon_U, \quad (7.30)$$

где k — весовой множитель, имеющий в системе СИ размерность Па/м^2.

Невязка ε_P вычисляется на участках поверхности, на которых заданы напряжения, а невязка ε_U — там, где заданы перемещения.

Выполняя дифференцирование обобщенного функционала по c_k, приходим к системе алгебраических уравнений, обеспечивающих его минимум:

$$\sum_{n=1}^{N} c_n \left[\frac{1}{S_P} \int_{S_P} \left(P_{xn} P_{xk} + P_{yn} P_{yk} + P_{zn} P_{zk} \right) dS_P + \right.$$

$$\left. + k \frac{1}{S_U} \int_{S_U} \left(U_{xn} U_{xk} + U_{yn} U_{yk} + U_{zn} U_{zk} \right) dS_U \right] =$$

$$= \frac{1}{S_P} \int_{S_P} \left(P_x^* P_{xk} + P_y^* P_{yk} + P_z^* P_{zk} \right) dS_P +$$

$$+ k \frac{1}{S_U} \int_{S_U} \left(U_x^* U_{xk} + U_y^* U_{yk} + U_z^* U_{zk} \right) dS_U. \quad (7.31)$$

Коэффициентом k можно регулировать погрешность удовлетворения граничным условиям. При увеличении k возрастает точность выполнения граничных условий в перемещениях и снижается в напряжениях. При уменьшении k эффект получается обратным. В практических вычислениях часто бывает удобно задавать

$$k = \left(E/L \right)^2, \quad (7.32)$$

где E — модуль упругости Юнга; L — характерный размер заданного тела.

Это соотношение следует из условия совпадения размерностей слагаемых в правой части уравнения (7.30).

Рассмотренное обобщение метода наименьших квадратов имеет недостаток, заключающийся в том, что погрешность удовлетворения граничным условиям на разных участках поверхности заданного тела распределяется неравномерно. Кроме того, в некоторых случаях бывает необходимо увеличить точность выполнения граничных условий на отдельных участках границы. Добиться требуемого эффекта можно, если в формулу (7.31) перед каждым интегралом ввести дополнительные безразмерные весовые коэффициенты k_m (где m — номер безразмерного коэффициента) и увеличивать значения весовых коэффициентов перед соответствующими граничными интегралами. Точность выполнения граничных условий на других участках при этом уменьшается.

Задача подбора необходимых значений весовых коэффициентов k_m является неоднозначной и представляет определенные трудности. В подразд. 7.3.4 излагается алгоритм ее решения, идея которого заимствована из искусственного интеллекта.

7.3. ИНТЕЛЛЕКТУАЛЬНЫЕ ПРОБЛЕМЫ МЕТОДА ФКО

Метод фиктивных канонических областей является аналитическим методом решения краевых задач и обладает характерным

для таких методов недостатком. Он плохо поддается алгоритмизации и может эффективно применяться только математиком-профессионалом, обладающим богатым опытом и хорошо развитой интуицией. Дело в том, что практически на всех стадиях решения краевой задачи возникают проблемы, для решения которых нет четких инструкций и правил. Ниже приводятся некоторые из основных проблем применения метода ФКО, а также способы их разрешения, основанные, главным образом, на идеях и методах искусственного интеллекта.

7.3.1. Прогнозирование особых точек решения

Из приведенных выше теоретических положений следует, что для успешного решения краевой задачи методом ФКО необходимо, во-первых, предсказать возможные места расположения особых точек искомого решения и, во-вторых, подобрать и расположить фиктивные канонические области так, чтобы они содержали область D и не содержали особые точки.

Вторая часть этой проблемы (выбор фиктивных канонических областей) решается путем применения рассмотренных выше методик суперпозиции ФКО и композиции расчетных областей. Предсказание же возможных мест расположения особых точек искомого решения краевой задачи является менее проработанной и наиболее сложной интеллектуальной проблемой применения метода ФКО. Дело в том, что точно указать особые точки функции, являющейся решением краевой задачи, можно только в том случае, если это решение известно. Таким образом, получается, своего рода, замкнутый круг. Приступая к решению краевой задачи методом ФКО нам надо заранее знать места расположения особых точек искомого решения. Однако определить их координаты можно только по виду решения краевой задачи, которое неизвестно.

При построении алгоритма, прогнозирующего возможные места расположения особых точек, будем руководствоваться следующими соображениями. Допустим, что $U(p)$ — это искомое точное решение краевой задачи, которое существует как в области D, так и за ее пределами, причем в точках ξ_1; ξ_2, ..., ξ_n, расположенных вне области D, функция $U(p)$ обращается в бесконечность. Функцию $U(p)$ можно визуализировать с помощью линий постоянного уровня (изолиний), а также линий, нормальных к изолиниям (показаны на рис. 7.10 штрихами). Как видно из рисунка, линии, нормальные к изолиниям искомого решения, сходятся в особых точках ξ_1, ξ_2, ..., ξ_n функции $U(p)$. Опираясь на подмеченное свойство особых точек, можно построить следующий алгоритм их прогнозирования [41].

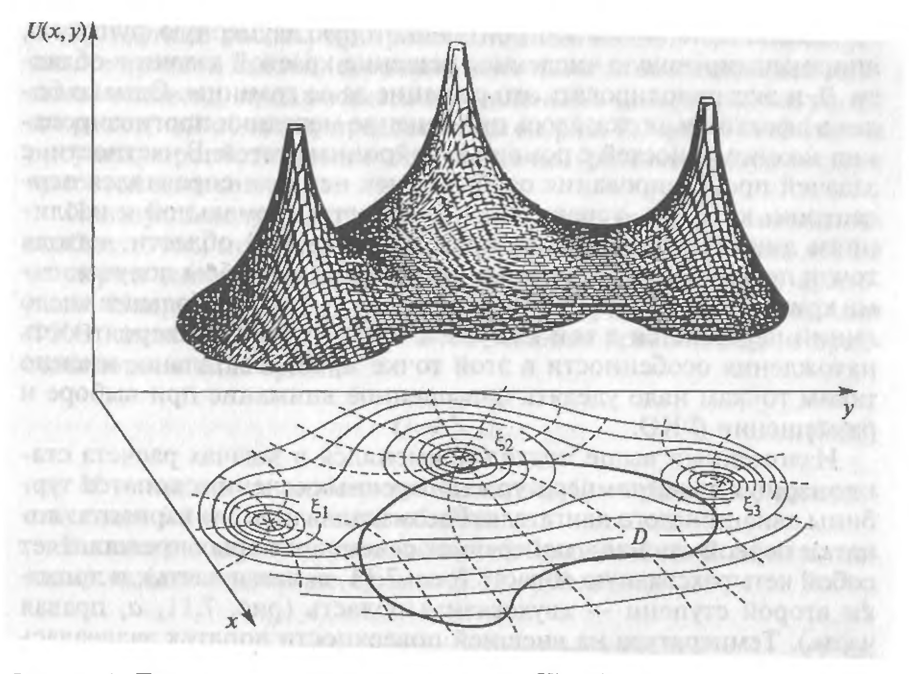


Рис. 7.10. Точное решение краевой задачи $U(x, y)$, которое существует, как в расчетной области D, так и за ее пределами, причем в точках ξ_1, ξ_2, ξ_3, расположенных вне области D, функция $U(x, y)$ обращается в бесконечность

1. Исходная краевая задача решается приближенно каким-либо численным методом.

2. В области D строится картина распределения линий, нормальных к изолиниям найденного решения.

3. Линии, нормальные к изолиниям, плавно продляются за границы расчетной области D. При этом они могут пересекаться в некоторых точках $\xi'_1, \xi'_2, ..., \xi'_n$, которые и принимаются за возможные места расположения особых точек искомого решения краевой задачи.

Первый шаг этого алгоритма может быть реализован любым численным методом, например, методом конечных разностей, конечных элементов, граничных элементов и т. д. При реализации второго шага удобно вычислять градиент функции $U(p)$, следуя вдоль которого легко построить линии, нормальные к линиям уровня. Наиболее затруднительным представляется третий шаг, поскольку задача продолжения построенных на предыдущем шаге нормальных линий за границы расчетной области D может быть решена однозначно только при наличии точного аналитического решения самой краевой задачи.

Можно попытаться построить некоторую двумерную функцию, аппроксимирующую численное решение краевой задачи в области D, и экстраполировать это решение за ее границы. Однако более эффективным оказалось применение методики прогнозирования закономерностей с помощью нейронных сетей. В частности, с задачей прогнозирования особых точек неплохо справлялся персептрон, который, усвоив закономерности нормальной к изолиниям линии, продлял ее за пределы расчетной области, находя точки пересечения с другими аналогичным способом полученными кривыми линиями. Естественно полагать, что чем большее число линий пересечется в той или иной точке, тем выше вероятность нахождения особенности в этой точке и, следовательно, именно таким точкам надо уделять повышенное внимание при выборе и размещении ФКО.

Изложенный выше подход применялся в задачах расчета стационарных полей температур в поперечных сечениях лопаток турбины авиационного двигателя. Рассматривались два варианта: лопатки первой ступени, поперечное сечение которых представляет собой четырехсвязную область (рис. 7.11, *а*, левая часть), и лопатки второй ступени — двухсвязная область (рис. 7.11, *а*, правая часть). Температура на внешней поверхности лопаток задавалась 1068 °С, на внутренних — 785 °С.

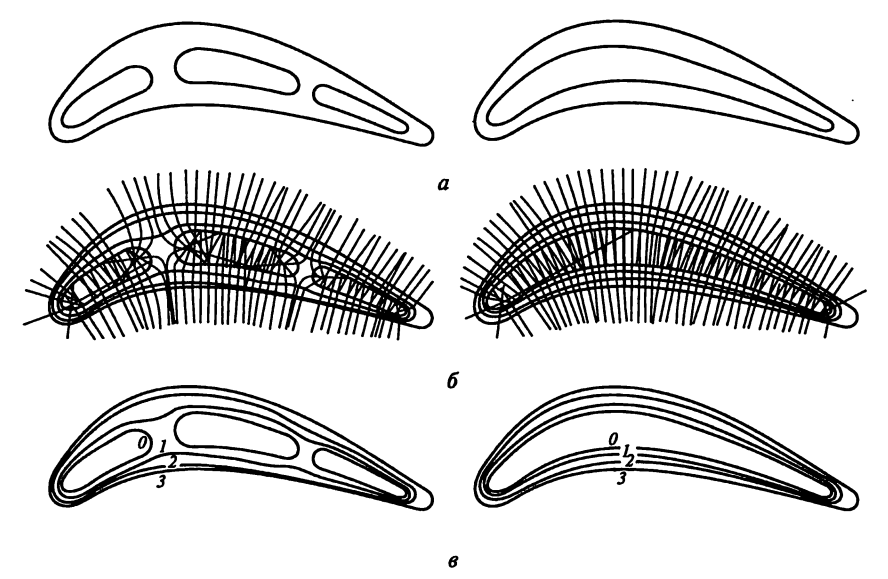


Рис. 7.11. Расчет температурных полей лопаток 1-й (слева) и 2-й (справа) ступеней турбины авиационного двигателя:

а — поперечное сечение лопаток; *б* — результат применения методики прогнозирования особых точек; *в* — картина распределения изотерм (*0* — 850 °С; *1* — 917 °С; *2* — 1002 °С; *3* — 1050 °С)

Результаты работы программы, выполняющей численное решение краевой задачи, строящей изолинии и имитирующей действия человека, продляющего нормальные к ним линии за границы расчетной области, представлены на рис. 7.11, *б*. Как видно из рисунка, прогнозируемые особые точки располагаются внутри полостей лопаток.

Пользуясь полученной прогностической картиной размещения особых точек, нетрудно подобрать и разместить ФКО так, чтобы их пересечение удовлетворяло сформулированным выше критериям. Например, можно воспользоваться приемом суперпозиции ФКО, согласно которому искомое решение краевой задачи представляется в виде суммы:

$$U = \sum_{i=1}^{M} U_i, \tag{7.33}$$

где M — число канонических областей; U_i — решение, соответствующее i-й канонической области.

В качестве канонических областей были выбраны кольца, поэтому каждое слагаемое суммы (7.33) имело вид

$$U_i = \sum_{n=-N}^{N} r_i^n (a_{ni} \cos n\varphi_i + b_{ni} \sin n\varphi_i) + c_i \ln r_i, \tag{7.34}$$

где r_i, φ_i — полярные координаты системы, центр которой совпадает с центром i-го кольца.

Для выполнения условий критериев выбора ФКО центры фиктивных колец были совмещены с найденными путем прогноза особыми точками.

В заключение отметим, что приведенная постановка задачи не отличается повышенной сложностью, учетом влияния различных физических факторов. С математической точки зрения она представляет собой задачу Дирихле для уравнения Лапласа в многосвязной области. Однако с помощью методики интеллектуального компьютерного моделирования эту краевую задачу удалось решить точно. Результаты решения приведены на рис. 7.11, *в* в виде распределения изотерм в расчетной области — поперечном сечении лопаток турбины авиационного двигателя.

7.3.2. Оптимизация расположения ФКО

Как отмечено ранее, успех решения каждой конкретной задачи методом ФКО (достижение малой невязки граничных уравнений) зависит от выбора и расположения фиктивных областей относительно заданного тела. Согласно критерию продолжимости выбирать ФКО следует так, чтобы область их пересечения содержала заданное тело D и не содержала особые точки искомого ре-

шения. Алгоритм прогнозирования особых точек, приведенный выше, облегчает задачу адекватного выбора ФКО. Однако такой прогноз является приближенным, поскольку, во-первых, исходным материалом для алгоритма прогнозирования является грубое приближенное решение краевой задачи, выполненное численным методом, а, во-вторых, сам алгоритм прогнозирования основан на эвристическом правиле и, как все системы искусственного интеллекта, дает заключение, не гарантирующее правильность. Кроме того, при известной информации о местах расположения особых точек возможно бесчисленное множество вариантов расположения ФКО, удовлетворяющих требованию критерия их выбора. Как показали результаты вычислительных экспериментов [42], все они различаются качеством получаемых решений — невязки удовлетворения граничным условиям могут изменяться от приемлемых значений до нуля. В связи с этим сформулируем задачу оптимизации расположения ФКО.

Пусть для тела D решается краевая задача, выбраны фиктивные области и их начальное расположение. Необходимо изменить положение ФКО с целью обеспечения наилучшего качества решения задачи. Рассмотрим алгоритм решения этой задачи, предложенный и реализованный С. Л. Гладким [4].

Качество решения определяется невязкой граничных уравнений, поэтому за критерий качества оптимизационной задачи можно принять значение граничного функционала. Пусть начальное

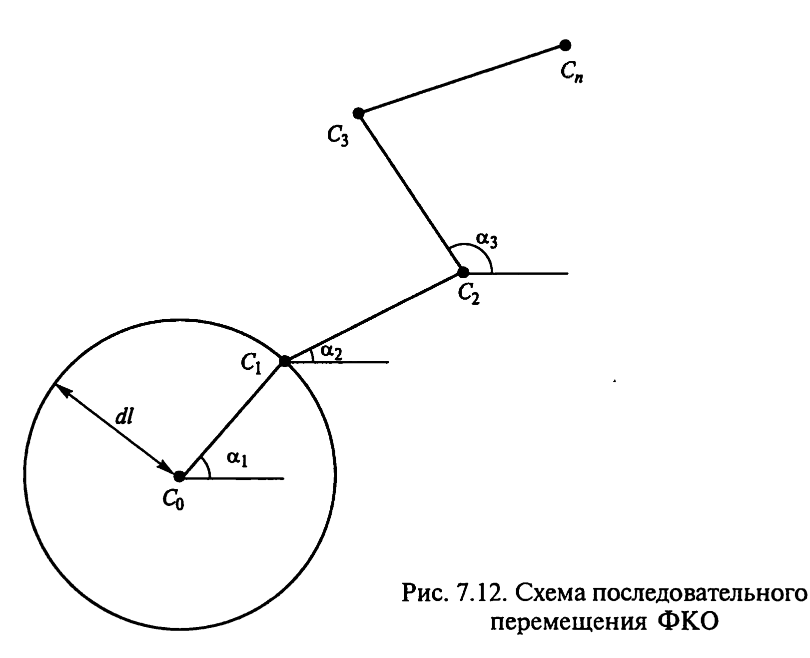


Рис. 7.12. Схема последовательного перемещения ФКО

положение некоторой ФКО определено положением ее центра C_0 (рис. 7.12). Рассматриваются все точки на расстоянии dl от C_0. Величина dl является входным параметром алгоритма и задается пользователем в долях характерного размера l тела D. Далее методом золотого сечения определяется направление перемещения (угол α_1) центра ФКО на расстояние dl, обеспечивающее уменьшение значения выбранного критерия качества. Центр ФКО помещается в найденную точку C_1, и вычисляется новое значение критерия качества. Если оно окажется меньше, чем в предыдущем положении, то центр ФКО остается в точке C_1 и далее осуществляются аналогичные действия. Если же значение критерия качества уменьшить не удалось, то величина шага dl уменьшается. Степень уменьшения величины dl также является входным параметром алгоритма. После нескольких итераций по одной ФКО она перемещается в некоторую точку C_n. Затем рассматриваются другие ФКО, для которых реализуются аналогичные итерационные алгоритмы, причем величина начального значения шага dl каждый раз восстанавливается. Процесс последовательного перемещения центров фиктивных областей происходит до тех пор, пока значение критерия качества не уменьшится в заданное число раз или общее число итераций не превысит максимально допустимое значение.

Численная реализация предлагаемого алгоритма выполнена так, что при оптимизации осуществляется динамическая визуализация хода решения, т.е. пользователь в реальном режиме времени получает полную информацию о каждой итерации — схему перемещения ФКО и график изменения значения граничного функционала. Если ход решения не устраивает пользователя, то он в любой момент может остановить итерационный процесс, изменить настройки алгоритма (значения входных параметров) и продолжить решение с того же места, на котором оно было остановлено, но с новыми параметрами.

Приведенный здесь итерационный динамически настраиваемый алгоритм оптимизации расположения ФКО реализован в программе REGIONS, предназначенной для решения краевых задач методом ФКО.

7.3.3. Распознавание плеонизмов

При решении краевых задач методом ФКО искомое решение задачи для заданной области D составляется из базисных функций, являющихся общим решением тех же дифференциальных уравнений, но относящихся к выбранным фиктивным каноническим областям. Для реальных задач часто оказывается так, что некоторые из выбранных даным способом функций не нужны для формиро-

вания решения краевой задачи для D. Например, если область D и заданные на ней граничные условия симметричны относительно оси x, то в базисном разложении (общем решении для V) не нужны функции, антисимметричные относительно оси x.

Другой пример. При использовании приема суперпозиции ФКО может получиться так, что некоторые функции, относящиеся к различным ФКО, будут линейно зависимы между собой. Если в первом случае наличие лишних функций в базисном разложении не приведет к существенному ухудшению качества получаемого решения, то во втором случае используемый метод решения разрешающей системы алгебраических уравнений может не справиться со своей задачей из-за имеющей место линейной зависимости между отдельными уравнениями. Лишние базисные функции, называемые *плеонизмами*, обычно стараются исключить на стадии подбора базисных разложений. Однако увидеть и распознать плеонизмы часто представляет собой сложную проблему, которую способен решить далеко не каждый математик.

Между тем, задача распознавания и исключения из сложной системы элементов, не влияющих или слабо влияющих на ее поведение, часто встречается в искусственном интеллекте. Так, в программах, моделирующих игру в шахматы (см. пподразд. 5.2), выявляются ходы, не оказывающие существенного влияния на развитие событий на шахматной доске. Это так называемые «мертвые» вершины дерева возможностей. При проектировании искусственных нейронных сетей (см. подразд. 3.3.2) выявляются и исключаются нейроны, не оказывающие влияние на решение, принимаемое сетью. В том и другом случаях используется прием, заключающийся в наблюдении за поведением характеристик системы при поочередном исключении ее элементов.

В случае решения краевой задачи методом ФКО такой характеристикой является среднеквадратичная погрешность удовлетворения краевым условиям ε. Если при исключении какой-либо базисной функции величина ε заметно увеличивается, то естественно полагать, что эта базисная функция необходима для формирования решения краевой задачи. В противном случае мы имеем дело с плеонизмом, который следует исключить.

Алгоритм, реализующий предлагаемую идею распознавания плеонизмов, может быть, например, таким. После определения коэффициентов c_n в базисном разложении (7.4) и определения среднеквадратичной погрешности удовлетворения краевым условиям ε исключается первое слагаемое ($c_1 = 0$) и вычисляется соответствующая этому случаю погрешность ε_1. Затем исключается второе слагаемое и так далее до $n = N$. Затем каждая ε_n сравнивается с ε. Если окажется, что $\varepsilon_n \leq \varepsilon$, то базисная функция $U_n(p)$ — плеонизм, подлежащий исключению.

7.3.4. Оптимизация весовых коэффициентов

При отыскании неизвестных коэффициентов общего решения методом наименьших квадратов (см. подразд. 7.2.3) вводятся весовые коэффициенты k_m, которые определяют значимость удовлетворения тому или иному виду граничных условий. Такие коэффициенты можно задать на любой границе и на любой ее части, а также для любой компоненты искомого решения краевой задачи. С помощью этих коэффициентов можно влиять на невязку удовлетворения граничным условиям на той или иной границе (участке границы).

Специфика метода граничных наименьших квадратов такова, что действует некий закон сохранения. Увеличивая какой-либо весовой коэффициент, мы уменьшаем невязку удовлетворения граничным условиям на соответствующей границе (участке границы). При этом невязки удовлетворения граничным условиям на других границах (участках границы) будут обязательно увеличиваться.

Математик, решающий краевую задачу, обычно стремится задать весовые коэффициенты так, чтобы невязки удовлетворения граничным условиям распределялись более-менее равномерно как по участкам границ, так и по компонентам искомого решения на границах. При этом он руководствуется интуитивными соображениями либо задает все коэффициенты k_m равными единице. Как правило, такие способы задания весовых коэффициентов редко приводят к равномерному распределению невязок.

Преследуя цель исключить творчество математика из процесса решения краевой задачи, можно предложить итерационный алгоритм поиска весовых коэффициентов, обеспечивающих равномерное распределение невязок. В основу такого алгоритма можно положить идею, аналогичную правилу Хебба (или дельта-правилу) для обучения персептрона (см. подразд. 3.1.3).

Суть алгоритма состоит в следующем. Весовым коэффициентам присваиваются начальные значения, равные единице, либо некоторые случайные значения. Решается краевая задача методом ФКО, в результате чего на каждом участке границы находятся невязки граничных уравнений (максимальные, среднеквадратичные или среднеинтегральные) ε_m, $m = \overline{1, M}$, где M — число участков границы. Затем вычисляются среднее значение невязки по всем границам $\varepsilon_c = \dfrac{1}{M}\sum_{m=1}^{M}\varepsilon_m$ и отклонения невязок на каждом участке от ε_c. Если на m-м участке границы окажется, что погрешность ε_m больше, чем ε_c, то соответствующее этому участку границы значение весового коэффициента k_m следует увеличить, в противном случае — уменьшить. Такая коррекция весовых коэффициентов

может осуществляться, например, с помощью итерационного процесса:

$$k_m(t+1) = k_m(t) + \eta(\varepsilon_m - \varepsilon_c) \tag{7.35}$$

или

$$k_m(t+1) = k_m(t)\frac{\varepsilon_m}{\varepsilon_c}, \tag{7.36}$$

где $k_m(t+1)$ и $k_m(t)$ — новое и старое значения весовых коэффициентов; η — коэффициент скорости обучения.

Итерации по формуле (7.35) либо (7.36) продолжаются до тех пор, пока отклонения всех невязок от ε_c не станут меньше заданной величины.

7.4. СИСТЕМА ИНТЕЛЛЕКТУАЛЬНОГО МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ REGIONS

Система REGIONS[1] предназначена для решения краевых задач математической физики с использованием метода ФКО. В ней реализованы все изложенные выше алгоритмы метода ФКО, в том числе и те, которые имитируют творческую деятельность математика-профессионала, решающего краевые задачи. Система позволяет получать точные аналитические решения плоских задач стационарной теплопроводности, теории упругости и термоупругости, осесимметричных задач стационарной теплопроводности и теории упругости.

Система написана на языке Object Pascal, имеет современный интерфейс и позволяет пользователю:

выбирать тип анализа. Предусмотрены следующие типы анализа: плоская задача теплопроводности, плоско-напряженное состояние, плоско-деформированное состояние, термоупругость — плоское напряжение и плоская деформация, осесимметричная задача теплопроводности, осесимметричное напряженное состояние;

задавать исходную область. Для построения области предусмотрены четыре вида графических примитивов: отрезок прямой линии, дуга окружности, дуга эллипса, сплайн (полином любого порядка);

проводить дискретизацию области. Каждую линию области можно разбить на произвольное число отрезков и задать коэффициент сгущения;

задавать граничные условия в зависимости от типа анализа. Для задач теории упругости можно задавать нормальные и касательные

---

[1] Система разработана С.Л. Гладким под руководством автора книги и помещена на сайте http://www.pspu.ru/regions.

к поверхности компоненты векторов напряжений и перемещений; для задач теплопроводности — граничные условия первого, второго и третьего рода. Для всех типов анализа предусмотрено задание условий симметрии. Все задаваемые параметры могут произвольно меняться вдоль любой линии. Также на каждой линии может быть задан свой коэффициент коррекции граничных уравнений;

вписывать заданную область в пересечение неограниченного числа ФКО. Для этого в системе есть несколько базовых типов ФКО, а также предусмотрена возможность создавать новые типы на их основе;

реализовывать метод композиции. Расчетная область может быть разделена на любое число подобластей, каждая из которых может быть погружена в пересечение любого числа ФКО;

решать поставленную задачу;

оценивать невязки удовлетворения граничным условиям, восстанавливать скорректированные граничные условия краевой задачи, которую удалось решить точно;

применять реализованные в пакете алгоритмы, имитирующие творческую деятельность математика-профессионала: прогнозировать особые точки, оптимизировать центры расположения ФКО, исключать плеонизмы, оптимизировать весовые коэффициенты базисных разложений;

писать программы на внутреннем языке программирования.

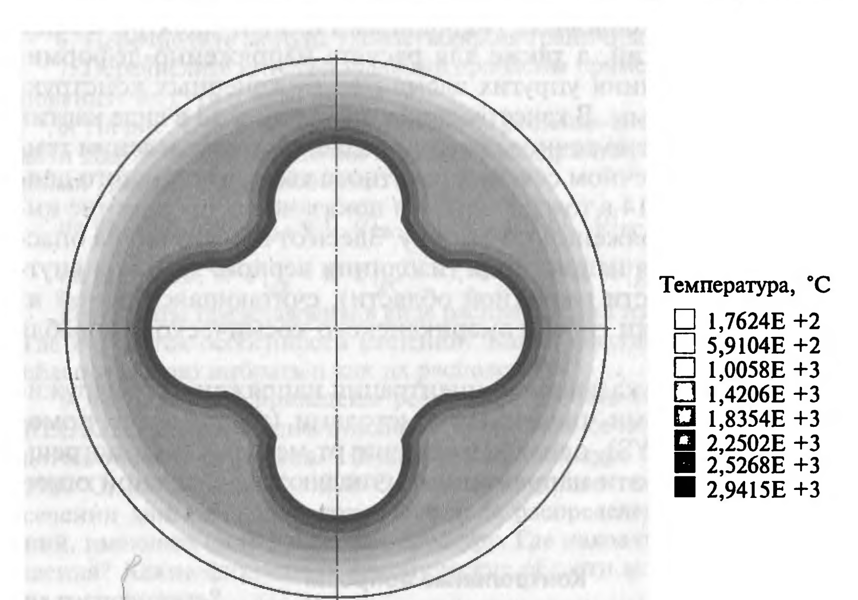


Рис. 7.13. Распределение температуры в поперечном сечении ракетного твердотопливного двигателя

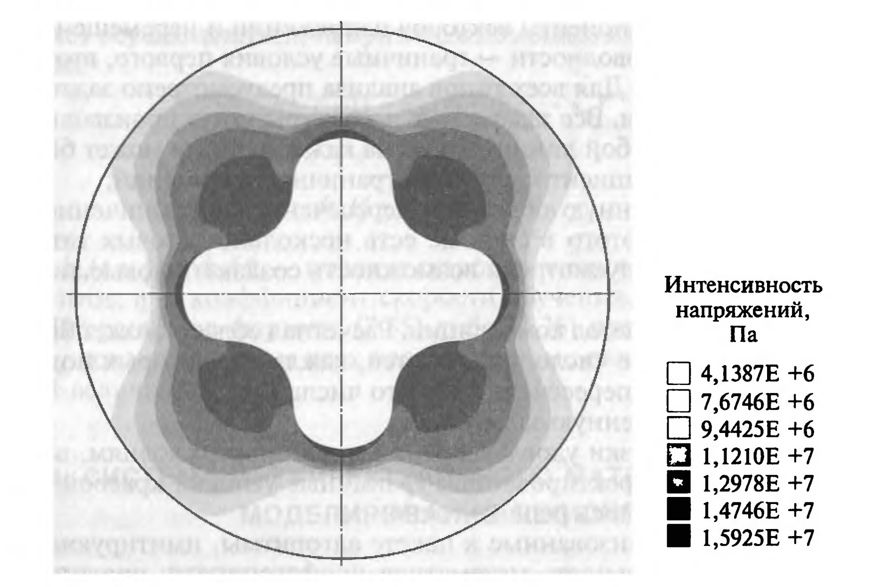


Рис. 7.14. Распределение интенсивности напряжений в поперечном сечении ракетного твердотопливного двигателя

Система REGIONS успешно применялась для компьютерного моделирования тепловых, гидродинамических, электрических, магнитных явлений, а также для расчета напряженно-деформированного состояния упругих элементов инженерных конструкций сложной формы. В качестве примера на рис. 7.13 в виде картин изолиний приведено точное решение задачи о распределении температуры в поперечном сечении ракетного твердотопливного двигателя. На рис. 7.14 в том же сечении показано распределение интенсивности напряжений по Мизесу. Здесь отчетливо видна опасная концентрация напряжений (изолинии черного цвета на внутренней поверхности расчетной области), считающаяся одной из возможных причин гибели американского космического корабля «Челленджер».

Отметим, что указанная концентрация напряжений обнаруживается и обычными численными методами (например, с помощью пакета ANSYS), однако, в отличие от метода ФКО, погрешность, с которой эти напряжения вычисляются, надежной оценке не поддается.

Контрольные вопросы

1. Чем отличаются численные методы решения краевых задач от аналитических? В чем заключается кризис современной прикладной математики?

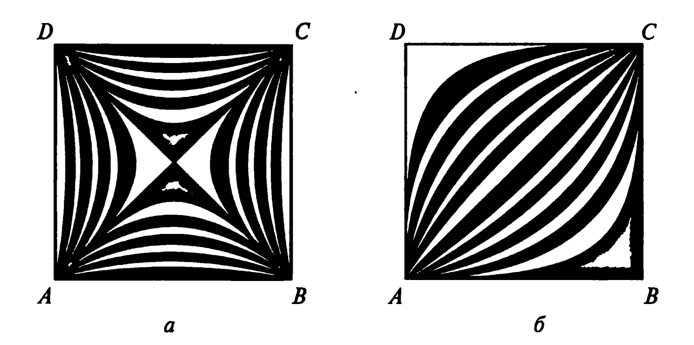


Рис. 7.15. Результаты решения задачи Дирихле для уравнения Лапласа, полученные с помощью системы REGIONS:

а, б — варианты распределения изолиний внутри квадратной области при различных граничных условиях

2. В чем состоит основное преимущество методов типа Треффтца перед другими аналитическими подходами?

3. В чем состоит идея метода фиктивных канонических областей?

4. В чем суть критерия продолжимости и из-за чего он может не выполняться?

5. Приведите пример, когда условие топологической эквивалентности выполняется, а условие продолжимости не выполняется. Приведите пример обратной ситуации.

6. Перечислите методы удовлетворения граничным условиям.

7. Перечислите интеллектуальные проблемы применения метода ФКО, поясните их суть и пути решения.

8. На рис. 7.15 приведены результаты решения системой REGION задачи Дирихле для уравнения Лапласа в квадратной области со следующими граничными условиями:

на рис. 7.15, *а*: $U|_{AB} = 1$; $U|_{BC} = 1$; $U|_{AD} = -1$; $U|_{DC} = -1$;

на рис. 7.15, *б*: $U|_{AB} = 1$; $U|_{BC} = -1$; $U|_{AD} = -1$; $U|_{DC} = 1$.

Результаты представлены в виде распределения изолиний функции U. Где находятся особенности решений? Какие фиктивные канонические области можно выбрать и как их расположить?

9. На рис. 7.13 приведены результаты расчета с помощью системы REGIONS температурного поля в поперечном сечении ракетного твердотопливного двигателя. На внутреннем контуре задана температура 2900 °C, на внешнем — 200 °C. Результирующее температурное поле в сечении двигателя представлено в виде распределения изотерм — линий, имеющих одинаковую температуру. Где находятся особенности решения? Какие фиктивные канонические области можно выбрать и как их расположить?

10. Зайдите на сайт http://www.pspu.ru/regions, ознакомьтесь с системой REGIONS и повторите решения краевых задач, результаты которых приведены на рис. 7.13—7.15.

СПИСОК ЛИТЕРАТУРЫ

1. *Арнольд В. И.* О функциях трех переменных // Доклады АН СССР. — М.: Изд-во АН СССР, 1957. — Т. 114. — № 4.

2. *Большаков А. Ю.*, *Елтышев В. А.* О решении пространственных задач теории упругости методом Фурье // Статические и динамические задачи упругости и вязкоупругости. — Свердловск: Изд. УНЦ АН СССР, 1983.

3. *Бордюже В. В.* Искусственные нейронные сети. — Пермь: Изд-во ПГТУ, 1999.

4. *Гладкий С. Л.*, *Ясницкий Л. Н.* Алгоритмы оптимизации базисных разложений в методе фиктивных канонических областей // Динамика и прочность машин. Вестник Пермского государственного технического университета. — Пермь, 2001. — № 3.

5. *Горбань А. Н.* Нейроинформатика и ее приложения // Открытые системы. — 1998. — № 04—06.

6. *Горбань А. Н.*, *Миркес Е. М.* Логически прозрачные нейронные сети. Нейроинформатика и ее приложения: Тезисы докладов III Всесоюзного семинара. — Красноярск: Изд-во КГТУ, 1999.

7. *Горбань А. Н.*, *Миркес Е. М.* Нейронные сети ассоциативной памяти, функционирующие в дискретном времени. Вычислительный центр СО РАН в г. Красноярске. Рукопись деп. В ВИНИТИ 17.07.97. № 2436-В97.

8. *Гусман С. Я.*, *Ясницкий Л. Н.* Обоснование выбора фиктивных канонических областей // Вестник Пермского университета. — Пермь: Изд-во ПГУ, 1994. — Вып. 1.

9. *Девингталь Ю. В.* Об оптимальном кодировании объектов при классификации их методами распознавания образов // Известия АН СССР. Техническая кибернетика. — 1968. — № 1.

10. *Девингталь Ю. В.* Кодирование объектов при использовании разделяющей гиперплоскости для их классификации // Известия АН СССР. Техническая кибернетика. — 1971. — № 3.

11. *Заенцев И. В.* Нейронные сети: Основные модели. — Воронеж: Изд-во Воронеж. ун-та, 1999.

12. Информатика: Учеб. пособие для студ. пед. вузов / Под ред. Е. К. Хеннера. — М.: Изд. центр «Академия», 2003.

13. Информатика: Учебник / Под ред. Н. В. Макаровой. — М.: Финансы и статистика, 1997.

14. Искусственный интеллект. Кн. 1. Системы общения и экспертные системы / Под ред. Э. В. Попова. — М.: Радио и связь, 1990.

15. Искусственный интеллект. Кн. 2. Модели и методы / Под ред. Д. А. Поспелова. — М.: Радио и связь, 1990.

16. Искусственный интеллект. Кн. 3. Программные и аппаратные средства / Под ред. В. Н. Захарова, В. Ф. Хорошевского. — М.: Радио и связь, 1990.

17. *Каллан Р.* Основные концепции нейронных сетей: Пер. с англ. — М.: Изд. дом «Вильямс», 2001.

18. *Киселев М.* Аналитические технологии: Средства добычи знаний в бизнесе и финансах // Открытые системы. — 1997. — № 4.

19. *Колмогоров А. Н.* О представлении непрерывных функций нескольких переменных суперпозицией непрерывных функций меньшего числа переменных // Доклады АН СССР. — М.: Изд-во АН СССР, 1956. — Т. 108.

20. *Колмогоров А. Н.* О представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения // Доклады АН СССР. — М.: Изд-во АН СССР, 1957. — Т. 114.

21. *Комарцова Л. Г., Максимов А. В.* Нейрокомпьютеры. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2002.

22. *Круглов В. В., Борисов В. В.* Искусственные нейронные сети. Теория и практика. — М.: Горячая линия — Телеком, 2002.

23. *Круглов В. В., Дли М. И.* Интеллектуальные информационные системы: Компьютерная поддержка систем нечеткой логики и нечеткого вывода. — М.: Изд-во физико-математической литературы, 2002.

24. *Кузин Л. Т.* Основы кибернетики. — М.: Энергия, 1979. — 584 с.

25. *Лейбензон Л. С.* Вариационные методы решения задач теории упругости. — М.—Л.: Гостехиздат, 1943.

26. *Люгер Дж. Ф.* Искусственный интеллект: Стратегии и методы решения проблем: Пер. с англ. — М.: Изд. дом «Вильямс», 2003.

27. *Минский М., Пейперт С.* Персептроны. — М.: Мир, 1971.

28. *Мкртчян С. О.* Нейроны и нейронные сети. Введение в теорию формальных нейронов. — М.: Энергия, 1971.

29. Нейроинформатика / А. Н. Горбань, В. Л. Дунин-Барковский, А. Н. Кирдин и др. — Новосибирск: Наука, 1998.

30. Нейрокомпьютеры и их применение. Кн. 5. Нейронные сети: история развития теории / Под ред. А. И. Галушкина и Я. З. Цыпкина. — М.: Радиотехника, 2001.

31. *Нильсон Н.* Обучающиеся машины. — М.: Мир, 1967.

32. *Нильсон Н.* Принципы искусственного интеллекта. — М.: Радио и связь, 1985.

33. Обработка знаний / Под ред. С. Осуга. — М.: Мир, 1989. — 293 с.

34. *Осовский С.* Нейронные сети для обработки информации: Пер. с польского. — М.: Финансы и статистика, 2002.

35. *Петров А. М., Мягких С. Г.* Из записной книжки полиграфолога. — Пермь: ИД «Компаньон», 2003.

36. *Попов Э. В.* Общение с ЭВМ на естественном языке. — М.: Наука, 1982.

37. Представление знаний / Под ред. С. Осуги, Ю. Саэки. — М.: Мир, 1990.

38. Представление и использование знаний / Под ред. Х. Уэно, М. Исидзука. — М.: Мир, 1987.

39. Распознавание образов / К. Вархаген, Р. Дейн, Ф. Грун, Й. Йостен, П. Вербек. — М.: Радио и связь, 1985.

40. *Слэйгл Дж.* Искусственный интеллект. — М.: Мир, 1973.

41. *Тарантина А. В., Ясницкий Л. Н.* Проблемы применения метода фиктивных канонических областей // Компьютерное и математическое моделирование в естественных и технических науках: Материалы IV Всерос. науч. internet-конф. (апрель-май 2002 г.). — Тамбов: ИМФИ ТГУ им. Г.Р.Державина, 2002. — Вып. 16.

42. *Тарантина А. В., Ясницкий Л. Н.* Влияние расположения особых точек искомого решения на сходимость метода фиктивных канонических областей. Численные иллюстрации // Динамика и прочность машин. Вестник ПГТУ. — № 4. — Пермь: Изд-во ПГТУ, 2003.

43. *Уинстон П.* Искусственный интеллект. — М.: Мир, 1980.

44. *Уоссермен Ф.* Нейрокомпьютерная техника. — М.: Мир, 1992.

45. *Уотермен Д.* Руководство по экспертным системам. — М.: Мир, 1989.

46. *Фогель Л., Оуэнс М.* Искусственный интеллект и эволюционное моделирование. — М.: Мир, 1969.

47. *Хант Э.* Искусственный интеллект. — М.: Мир, 1978.

48. *Эндрю А.* Искусственный интеллект. — М.: Мир, 1985.

49. *Ясницкий Л. Н.* Об одном способе решения задач теории гармонических функций и линейной теории упругости // Прочностные и гидравлические характеристики машин и конструкций. — Пермь: Изд-во Пермского политехн. ин-та, 1973.

50. *Ясницкий Л. Н.* Метод фиктивных канонических областей в механике сплошных сред. — М.: Наука, 1992.

51. *Ясницкий Л. Н.* Суперпозиция базисных решений в методах типа Треффтца // Изв. АН СССР. Механика твердого тела. — 1989. — № 2.

52. *Ясницкий Л. Н.* Композиция расчетной области в методе фиктивных канонических областей // Изв. АН СССР. Механика твердого тела. — 1990. — № 6.

53. *Cover T.* Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition // IEE Trans. Electronic Computers, 1965. — V. 14.

54. Haykin S. Neural Networks: A Comprehensive Foundation // MacMillan College Publishing Co. New York, 1994.

55. *Hebb D. O.* The Organization of Behavior. John Wiley & Sons, New York, 1949.

56. *Hecht-Nielsen R.* Kolmogorov's Mapping Neural Network Existence Theorem // IEEE First Annual Int. Conf. On Neural Networks. San Diego, 1987. — V. 3.

57. *Hecht-Nielsen R.* Neurocomputing. — Amsterdam: Addison Wesley, 1991.

58. *Holland J. H.* Adaptation in Natural and Artifical Systems. University of Michigan Press, 1975.

59. *Hopfield J. J.* Neurons with graded response have collective computational properties like thous of two-state neurons. Proceedings of the National Academy of Sciences, 1984. — № 81.

60. *Klimauskas G.* Neural Ware — User manual. Natick, USA: Neural Ware Inc., 1992.

61. *Kohonen T.* Self-organizing maps. — Berlin: Springer Verlag, 1995.

62. *McCulloch W. S., Pitts W.* A Logical Calculus of Ideas Immanent in Nervous Activity // Bull. Mathematical Biophysics, 1943. — V. 5.

63. *Rosenblatt F.* The perseptron: a probabilistic model forinformation storage and organization in the brain // Psychological Review, 1958. — V. 65.
64. *Rosenblatt F.* Principles of Neurodynamics. Spartan Books.— New York, 1962.
65. *Rummelhart D. E., Hilton G. E., Williams R. J.* Learning internal representations by error propagation. In McClelland et al., 1986.
66. *Selfridge. O.* Pandemonium. A paradigm for learning. Proceedings of Symposium on Mechanization of Thought Processes. — London, 1959.
67. *Werbos P.* Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences // Phd Thesis, Dept. of Applied Mathematics. — Harvard University, Cambridge, Mass., 1974.
68. *Widrow B., Hoff M. E.* Adaptive swiching circuits. 1960 IRE WESTCON Conferenction Record. — New York, 1960.
69. *Widrow B., Lehr M.A.* 30 years of adaptive neural networks: perseptron, madaline and backpropagation // Proceedings of the IEEE, 1990. — V. 78. № 9.

ОГЛАВЛЕНИЕ

Предисловие ... 3

Глава 1. **Прошлое, настоящее и будущее искусственного интеллекта** ... 6
1.1. Исторический очерк ... 6
1.2. Направления развития искусственного интеллекта ... 10

Глава 2. **Системы, основанные на знаниях** ... 14
2.1. Данные и знания ... 14
2.2. Методы представления знаний ... 15
2.2.1. Продукционные правила ... 15
2.2.2. Фреймы ... 18
2.2.3. Семантические сети ... 19
2.3. Экспертные системы ... 20
2.3.1. Предметные области ... 20
2.3.2. Обобщенная структура ... 20
2.3.3. Этапы и технология разработки ... 22

Глава 3. **Нейроинформатика** ... 26
3.1. Персептрон и его развитие ... 26
3.1.1. Мозг и компьютер ... 26
3.1.2. Математический нейрон Мак-Каллока—Питтса ... 28
3.1.3. Персептрон Розенблатта и правила Хебба ... 30
3.1.4. Дельта-правило и распознавание букв ... 32
3.1.5. Адалайн, мадалайн и обобщенное дельта-правило ... 34
3.1.6. Ограниченность однослойного персептрона ... 37
3.1.7. Многослойный персептрон и алгоритм обратного распространения ошибки ... 39
3.2. Возможности и области применения персептронов ... 45
3.2.1. Новый подход к методу математического моделирования ... 45
3.2.2. Диагностика в медицине ... 46
3.2.3. Диагностика неисправностей сложных технических устройств ... 47
3.2.4. Нейросетевой детектор лжи ... 50
3.2.5. Нейросеть-антихакер ... 51
3.2.6. Нейросети в банковском деле ... 52
3.2.7. Прогнозирование валютных курсов и котировок ценных бумаг ... 53
3.2.8. Задачи, решаемые с помощью нейросетей ... 56
3.2.9. Невербальность и «шестое чувство» нейросетей ... 57

3.3. Проектирование и обучение персептронов 60
3.3.1. Теоремы существования 60
3.3.2. Проблемы и методы проектирования 61
3.3.3. Проблемы и методы обучения 65
3.3.4. Подготовка входных и выходных параметров 73
3.3.5. Виды активационных функций 76
3.4. Радиально-базисные сети 78
3.5. Рекуррентные сети 82
3.5.1. Рекуррентные сети на базе персептрона 82
3.5.2. Сеть Хопфилда 84
3.6. Самообучающиеся и гибридные сети 87

Глава 4. **Распознавание образов** 93
4.1. Проблема распознавания образов 93
4.2. Пандемониум Селфриджа 94
4.3. Персептрон Розенблатта 97
4.4. Распознавание символов 99
4.4.1. Методы распознавания символов 99
4.4.2. Предварительная обработка изображений 100
4.4.3. Распознавание по методу Паркса 103
4.4.4. Современные системы распознавания текстов 105
4.5. Использование геометрических интерпретаций 110

Глава 5. **Интеллектуальные игры** 115
5.1. Понятия игры и дерева возможностей 115
5.2. Методы подрезки дерева возможностей 116
5.3. Идеи обучения игровых программ 121

Глава 6. **Компьютерное творчество** 126
6.1. Философские аспекты творчества 126
6.2. Моделирование в музыке 130
6.3. Моделирование в поэзии 133

Глава 7. **Интеллектуальное математическое моделирование** 137
7.1. Современный кризис прикладной математики 137
7.2. Метод фиктивных канонических областей 141
7.2.1. Идея и теоретические основы 141
7.2.2. Иллюстрации на тестовой задаче и другие правила 148
7.2.3. Способы удовлетворения краевым условиям 153
7.3. Интеллектуальные проблемы метода ФКО 157
7.3.1. Прогнозирование особых точек решения 158
7.3.2. Оптимизация расположения ФКО 161
7.3.3. Распознавание плеонизмов 163
7.3.4. Оптимизация весовых коэффициентов 165
7.4. Система интеллектуального математического моделирования REGIONS 166

Список литературы 170

Учебное издание

Ясницкий Леонид Нахимович

Введение в искусственный интеллект

Учебное пособие

2-е издание, исправленное

Редактор *Е. М. Зубкович*
Технический редактор *Н. И. Горбачева*
Компьютерная верстка: *Л. А. Смирнова*
Корректор *Т. В. Кузьмина*

Изд. № 102107207. Подписано в печать 15.05.2008. Формат 60×90/16.
Гарнитура «Таймс». Печать офсетная. Бумага офсетная № 1. Усл. печ. л. 11,0.
Тираж 1 500 экз. Заказ № 26587.

Издательский центр «Академия». www.academia-moscow.ru
Санитарно-эпидемиологическое заключение № 77.99.02.953.Д.004796.07.04 от 20.07.2004
117342, Москва, ул. Бутлерова, 17-Б, к. 360. Тел./факс: (495) 330-1092, 334-8337.

Отпечатано в полном соответствии с качеством диапозитивов, предоставленных издательством в ОАО «Саратовский полиграфкомбинат».
410004, г. Саратов, ул. Чернышевского, 59. www.sarpk.ru.